MOSE: A Monotonic Selectivity Estimator Using Learned CDF

Luming Sun, Cuiping Li*, Tao Ji, and Hong Chen

Abstract—The accuracy of selectivity estimation is of vital importance to create good query plans. Traditional estimators such as histograms make several assumptions during estimation that can lead to huge errors. Recently the database community started exploring the usage of machine learning in selectivity estimation and won great achievements. However, due to the black box models they used, existing learning-based methods still face a variety of new challenges, including high estimation latency, large training data demanding, and occurrence of illogical results. In this work, we propose a learning-based **MO**notonic **S**electivity **E**stimator (MOSE) to address these challenges. We first learn a multi-dimensional **C**umulative **D**istribution **F**unction (CDF) of the data in a supervised method and then compute selectivity for ad hoc query predicates at rum-time. We propose a novel regularizer and an effective attribute-aware calibration method to improve the estimation accuracy. To further improve the model efficiency, we design a mutual information based attributes partition algorithm for model ensemble. With regard to the heavy cost of training data collection, we design a model-based active learning strategy to generate high-quality training data cost-effectively. We conduct extensive experiments on both real-world and synthetic datasets and the results show that MOSE outperforms the state-of-the-art methods in terms of accuracy and efficiency.

Index Terms—Selectivity Estimation, Machine Learning, Query Optimization

1 Introduction

Selectivity estimation is one of the key components of query optimization, which aims to estimate the fraction of tuples in a database that satisfy the query predicate [1]. These estimates are used by the cost-based query optimizer to choose an optimized query plan [2]. However, recent studies show that the traditional selectivity estimation methods such as histograms [3], [4] and sampling [5], [6] often generate poor estimates, causing unexpectedly bad query performance [1].

Machine learning has been widely adopted in many different applications in the last decade. The database community also has started to explore how machine learning techniques can be leveraged in database management systems [7], [8]. Several machine learning-based methods have been proposed to improve the performance of multi-dimensional selectivity estimation without unreliable assumptions in reality such as attribute independence [9], [10], [11], [12]. Although these models are able to capture attribute correlation much better, they face several new challenges. For example, the lightweight models [9] and QuickSel [12] treat selectivity estimation as a supervised machine learning task. But such *query-based* methods require plenty of training queries in order to make accurate predictions. However, collecting such training samples is very time-consuming [9], [12], [13]. Naru [10] and DeepDB [11] are another kind of learned selectivity estimators. Such methods learn the joint data distribution with unsupervised machine learning models. Although such *data-based* methods can capture conditional data distribution quite well, it is still unpractical to deploy these models in a production scenario due to their heavy costs on model training and predicting [14]. Besides, most existing learned estimators violate some or all of the following basic rules in selectivity estimation [14]:

- **Monotonicity:** A larger range predicate should have a larger or equal selectivity.
- **Validity:** Results of invalid queries such as SELECT * FROM T WHERE $1 \leq A \leq 0$ should be 0.
- **Consistency:** Result of a range predicate should be equal to the sum of results of ranges splitted from it (e.g. selectivity of $0 \leq A \leq 2$ should be equal to sum of $0 \leq A < 1$ and $1 \leq A \leq 2$).
- **Stability:** The predicted selectivity from an estimator should always be stable for the same query when data is not changed.

To address the above challenges, in this paper, we propose a learning-based monotonic selectivity estimator, MOSE, aiming for accurate, reliable, and efficient selectivity estimation. Our key observation is that the joint cumulative distribution function of the data in a table can be used to compute the selectivity for ad hoc query range predicates. This approach turned out to be very powerful and selectivity can be computed in constant time regardless of the size of the query range. To this end, we first propose a novel technique based on lattice regression model [15] to learn multi-dimensional CDF of the data in a query-based fashion. In order to make the learned CDF fit the distribution better and avoid over-fitting, we propose a cell-wise regularizer that penalizes parameters in the model with different weights relevant to the data distribution. And we also propose a new method to calibrate the range features with attribute-aware information for estimation accuracy. To further improve the

- *Luming Sun, Cuiping Li, Tao Ji, and Hong Chen are with the Key Laboratory of Data Engineering and Knowledge Engineering, Ministry of Education, China, and School of Information, Renmin University of China, China. (E-mail: sunluming,licuiping,jitao,chong@ruc.edu.cn). (Corresponding author: Cuiping Li.)*

efficiency of the method, we design a mutual information based attributes partition algorithm for model ensemble. Facing the common issue of heavy cost on training data collecting, we design an active data generator and propose a model-based active learning algorithm to generate high-quality training data cost-effectively. We identify our contributions as follows.

1) We present novel techniques to estimate selectivity for queries with range predicates efficiently and reliably. The main idea is to train a query-based CDF learner for multi-dimensional data. Then the selectivity of any range predicates can be computed by accessing and combining appropriate cumulative probabilities.
2) In order to adapt the lattice regression model to CDF learning and improve the accuracy and usability of the CDF learner, we propose a cell-wise regularizer and an attribute-aware calibration method. And we adopt model ensemble with a mutual information based attributes partition algorithm to improve efficiency.
3) We design an active data generator to generate high-quality training data cost-effectively. The proposed active learning algorithm is deeply integrated with our lattice model for CDF learning.
4) We conducted extensive experiments on two real-world datasets and one synthetic benchmark with both range and categorical predicates. Compared to the state-of-the-art query-based selectivity estimation method, MOSE achieves up to 62% less error with one-fifteenth parameters and achieves a 3.29x speedup on selectivity estimation. Moreover, compared with random data generation, our active data generator helps provide roughly equal quality estimation with up to 50% fewer training samples.

2 Problem Formulation

We first give relevant notations and definitions in Section 2.1 and then describe how to compute the selectivity with the multi-dimensional CDF in Section 2.2. Next, in Section 2.3, we formulate the CDF learning as a regression problem.

2.1 Notations and Definitions

Table 1 lists the main symbols we use throughout the paper. The relational table T of a database is composed of d *attributes* $A_1, \ldots, A_d$, and we denote the domain of an attribute A_i as $dom(A_i)$. We represent T as a d-dimensional space D with starting index 1. Assume the total number of tuples of T is N. Each dimension of D is the *rank domain* of a corresponding attribute of T. Thus, the attributes of T need not be restricted to the continuous integer domain. A point x in space D is represented by a vector x=$[x_1, x_2, \ldots, x_d]$, where $x_i \in [0, |dom(A_i)|]$.

Range Predicates. There are usually filter predicates in SQL *WHERE* clauses, and we refer to the range constraints on numerical attributes as range predicates. A range predicate p is a conjunction of *range constraints* $R = R_1 \times R_2 \times \ldots \times R_d$, where $R_i = [l_i, u_i]$ is the range constraint on the i^{th} dimension of D. Geometrically, R is the area bounded by a hyperrectangle whose bottom-left corner (lower bound) is $(l_1, l_2, \ldots, l_d)$ and top-right corner (upper bound) is $(u_1, u_2, \ldots, u_d)$.

TABLE 1: Symbols

Symbol	Definition and Description
T	relational table
N	the number of tuples in T
A_i	the i^{th} attribute of table T
R	the query range
R_i	the range constraint on the i^{th} attribute
D	multi-dimensional space mapped from table T
C^i	the lattice size of the i^{th} attribute
C_i	the i^{th} node of the lattice
d	the number of attributes of table T
p	the query predicate on table T
x	the point in space D, $x \in \mathcal{R}^d$
l_i	the lower bound of the i^{th} range constraint
u_i	the upper bound of the i^{th} range constraint
n	the number of training instances
m	the number of nodes in the lattice
c	the cell of the lattice

Selectivity. Given a range predicate p with query range $R = [l_1, u_1] \times [l_2, u_2] \times \ldots \times [l_d, u_d]$, the selectivity of p (denoted as $sel(p)$) is defined as the *fraction* of the tuples that satisfy the predicate p. That is,

$$sel(p) = \frac{1}{N} \sum_{i=1}^{N} I(t_i \in R) \,, \tag{1}$$

where $I(\cdot)$ is the indicator function and $I(\cdot) = 1$ if tuple t_i of T satisfies the predicate p (i.e., t_i is contained by the hyperrectangle R). Note the concept of cardinality in database is closely related to selectivity, and it can be computed by $N \times sel(p)$.

Multi-Dimensional CDF. The cumulative distribution function (CDF) of a d-dimensional real-valued random variable $X = (X_1, X_2, \ldots, X_d)$ is defined as $F_X(x)$=$Pr(X_1 \leq x_1, X_2 \leq x_2, \ldots, X_d \leq x_d)$, where x=$[x_1, x_2, \ldots, x_d]$ is a point of multi-dimensional space D. It can be computed as:

$$F_X(x) = \frac{1}{N} \sum_{i=1}^{N} I(x' \in R') \,, \tag{2}$$

where x' is a point in space D and $R' = [0, x_1] \times [0, x_2] \times \ldots \times [0, x_d]$.

2.2 CDF to Selectivity

Given the CDF, Our key observation is that the exact selectivity of any range predicates can be computed by accessing and combining 2^d appropriate cumulative probabilities, where d is the number of numeric dimensions for which ranges have been specified in the query. The theorem below (motivated by [16]) provides how selectivity can be computed from up to 2^d appropriate cumulative probabilities. For notational convenience, let $F_X(x)$ = 0 if x_j = -1.

Theorem 1. Given a predicate p with query range $R = [l_1, u_1] \times [l_2, u_2] \times \ldots \times [l_d, u_d]$ on d dimensions, then for any $j \in \{1, 2, \ldots, d\}$, let

$$s(j) = \begin{cases} -1, & x_j = l_j - 1, \\ 1, & x_j = u_j. \end{cases}$$

Then, the selectivity of p can be computed as:

$$sel(p) = \sum_{\forall x_i \in \{l_i - 1, u_i\}} \left\{ \left(\prod_{i=1}^{d} s(i) \right) F_X(x) \right\} . \quad (3)$$

The proof of Theorem 1 is similar to the basic range-sum algorithm in data cube [16], and the details are omitted to conserve space. For example, when $d = 2$, $sel(p)$ for predicate p with query range $R = [l_1, u_1] \times [l_2, u_2]$ can be obtained as:

$$sel(p) = F_X([u_1, u_2]) - F_X([u_1, l_2 - 1]) - F_X([l_1 - 1, u_2]) + F_X([l_1 - 1, l_2 - 1]) .$$

Figure 1 demonstrates the geometrical explanation of above computation steps. Each rectangle in the figure stands for the domain area of the two attributes, and the shaded area represents the selectivity of the covered space.

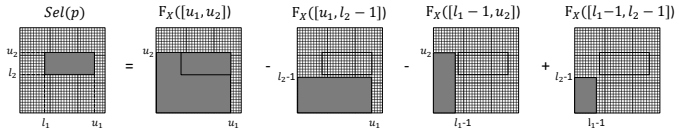


Fig. 1: Transformation from CDF to selectivity

2.3 Formulation as a Regression Problem

Theorem 1 shows that once we obtained the multi-dimensional CDF of the data, the selectivity for any query predicate can be computed in constant time regardless of the size of the query range. However, the computation of CDF is non-trivial since the time complexity for computing the cumulative probability of any point x is $O(N * d)$ according to Equation 2. The computation will take very a long time to complete especially when the size of N and d are large. This latency is unacceptable in most real environments.

To improve efficiency, in this paper, instead of computing the exact CDF, we propose a novel technique to learn high-quality CDF in a query-based fashion. Key to this goal is that the cumulative probability of point x=$[x_1, x_2, \ldots, x_d]$ is actually the selectivity of query with range of $[0, x_1] \times [0, x_2] \times \ldots \times [0, x_d]$ (Equation 2); we can use such kinds of queries as training instances and formulate CDF learning as a supervised regression problem.

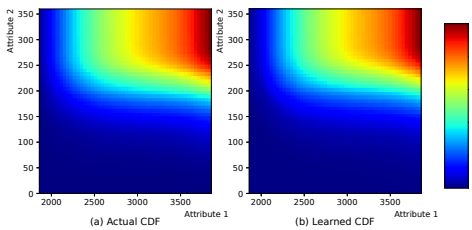


Fig. 2: Joint CDF two attributes from Forest dataset

Problem (Query-based CDF Learning) Given a set of n observed queries $(p_1, sel(p_1)), \ldots, (p_n, sel(p_n))$ for T, where $p_i (1 \leq i \leq n)$ is range predicates with lower bound [0, ..., 0], and given a new query predicate p_{n+1} with query range $R = [0, x_1] \times [0, x_2] \times \ldots \times [0, x_n]$, our goal is to build a model that can estimate the selectivity $sel(p_{n+1})$. please note that $sel(p_{n+1})$ is just the cumulative probability of point x. Since x can be any point in D, thus by this way we can learn the whole CDF through the model.

Figure 2(a) shows the true CDF computed using the first two attributes of Forest dataset [17], and the cumulative probability value is showed in different colors. It's obvious that multi-dimensional CDF is monotonically non-decreasing with each of its dimensions. Figure2(b) shows the learned CDF by our model. It can be easily found that our learned CDF guarantees monotonicity and is highly consistent with the actual CDF.

Support Queries We support queries with range, equality and IN predicates on multiple numeric and categorical attributes. Currently, we focus on selectivity estimation of a single table, but our method can be extended to queries with join predicates by materializing joins as a base table like [18].

3 System Overview

In this section, we present the system overview of our monotonic selectivity estimator. Selectivity estimation really ultimately comes down to modeling the probability distribution of underlying data, so we go to the root of the problem and propose a monotonic selectivity estimator with clear interpretability in terms of accumulative probability.

Figure 3(a) demonstrates how our method for selectivity estimation can be integrated into typical database systems. From an overall perspective, the monotonic selectivity estimator plays the same role as traditional estimators such as histogram, and the only difference is that we need query feedback (i.e. the actual selectivity) after the queries are executed. The proposed estimator consists of four main components as follows:

1) CDF Learner Different from existing query-based methods that directly estimate selectivity from a model, MOSE first learns the multi-dimensional CDF based on the data pool (Section 4).

2) Data pool Training queries composed of three parts are added to the data pool with the selectivity feedback after they are executed.

3) Active data generator In order to obtain the most valuable training samples and avoid the heavy cost of data collection, a data generator using a novel active learning strategy is designed for our CDF learner. (Section 5)

4) Selectivity Computer After the CDF learner is trained, the selectivity for any query predicate can be computed by using Theorem 1.

Workflow. During offline training, the *CDF learner* is firstly trained with randomly generated initial training data. Then the *Active data generator* generates representative training queries with our lattice-based active learning strategy (Section 5). After these new queries are executed by the executor, the query-selectivity pairs are added into the *Data pool* as active generated set. The *CDF learner* then updates its parameters with all instances from *Data pool*. There is a loop between the *Data pool*, the *CDF learner*, and the *active data generator*, meaning that there can be several iterations of training-generating process until the total cost exceeds a specified threshold.

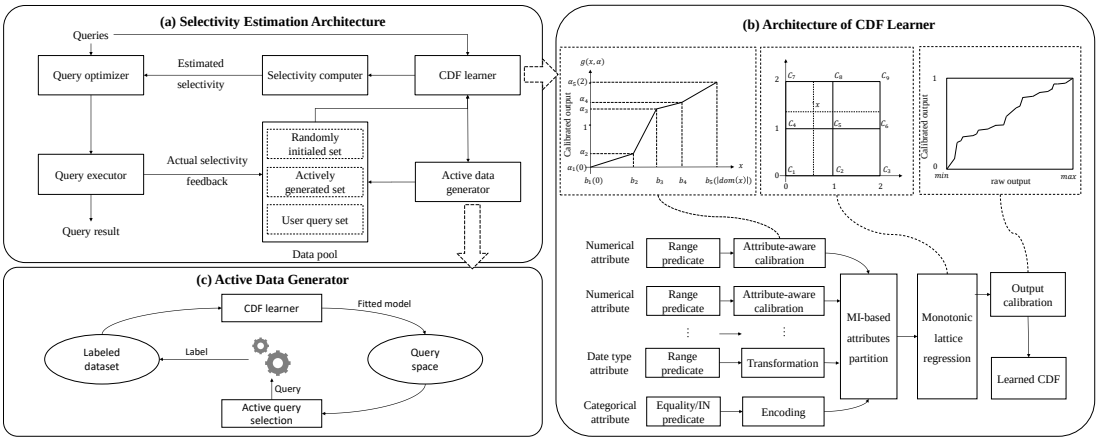


Fig. 3: System overview

For online selectivity estimation, when the query optimizer demands selectivity of one predicate, MOSE calculates the selectivity based on the trained *CDF Learner* with Equation 3. Then the estimated result is fed to the query optimizer. After the query is executed, the *Data Pool* collects the actual selectivity feedback as user query set. The *CDF Learner* can be fine-tuned with the users' queries for better performance.

$$\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \\ \theta_8 \\ \theta_9 \end{bmatrix} \leq \mathbf{0} \Leftrightarrow \begin{matrix} \theta_1 \leq \theta_2 \\ \theta_2 \leq \theta_3 \\ \theta_4 \leq \theta_5 \\ \theta_5 \leq \theta_6 \\ \theta_7 \leq \theta_8 \\ \theta_8 \leq \theta_9 \\ \theta_1 \leq \theta_4 \\ \theta_2 \leq \theta_5 \\ \theta_3 \leq \theta_6 \\ \theta_4 \leq \theta_7 \\ \theta_5 \leq \theta_8 \\ \theta_6 \leq \theta_9 \end{matrix}$$

Fig. 4: Monotonicity constraints

4 Monotonic CDF Learner

To learn the joint CDF from queries while maintaining monotonicity, we adopt the monotonic lattice regression model in our CDF learner. We first introduce the monotonic lattice regression model in Section 4.1, and then we discuss how to adapt the lattice regression model to CDF learning task with a cell-wise regularizer (Section 4.2), an attribute-aware calibration function(Section 4.3), and model ensemble (Section 4.4). Finally, we present the model extension for categorical predicates in Section 4.5.

4.1 Monotonic Lattice Regression

The actual joint CDF is monotonically increasing with each attribute, but common regression models such as neural networks can't promise a larger output as the input features increase. Recently proposed monotonic lattice regression [15] addressed the monotonicity issue by adding linear inequality constraints to optimization functions, thus guaranteeing the monotonicity between output predictions and input features.

The basic idea of monotonic lattice regression is first to estimate an intermediate function from the given samples using a regression method, and pre-compute the values of an intermediate function at the nodes of a regular lattice and store those values. Then, given a new input sample x that lies within the domain of the lattice, the estimated function $f(x)$ is evaluated by interpolating x from its surrounding lattice nodes.

In our setting, monotonic lattice regression constructs regular lattice over space D. Let C^i be a hyperparameter specifying the number of lattice nodes (i.e. lattice size) of the i^{th} dimension. Then the lattice L is a regular grid with m lattice nodes $\{C_i \in Z^d\}_{i=1:m}$ and $m = C^1 \times C^2 \times \ldots \times C^d$. For example, the monotonic lattice regression in Figure 3(b) shows a 3×3 lattice on 2 dimensions and $[C_1, C_2, \ldots, C_9]$ are lattice nodes. We call hyperrectangles in lattice L as cells, and each cell c is bounded by 2^d lattice nodes. In fact, the lattice L constructs grids that are overlaid on space D. Thus a cell of L contains a subset of points of D.

For any point x of D, its estimated cumulative probability $\hat{y} = F_X(x)$ is calculated by linearly interpolating the lattice parameter θ_j $(1 \leq j \leq m)$ for the cell containing x:

$$\hat{y} = F_X(x) = \sum_{j=1}^{m} \phi(x)_j \theta_j,$$

where $\phi(x)$ is the linear interpolation weights that satisfy the following equations:

$$\sum_{j=1}^{m} \phi(x)_j C_j = x, \quad \sum_{j=1}^{m} \phi(x)_j = 1.$$

Given a training set with n samples pairs $(p_i, sel(p_i))$, where the upper bound of the query range of p_i is $(a_1, a_2, \ldots, a_d)$ and $sel(p_i) = F_X([a_1, a_2, \ldots, a_d])$ (Equation 2), the lattice parameter $\theta = [\theta_1, \theta_2, \ldots, \theta_m]$ can be obtained by minimizing the empirical error between the true cumulative probability y and the estimated one $\hat{y}$:

$$\begin{aligned} \theta &= \arg\min_{\theta \in \mathbb{R}^m} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \\ &= \arg\min_{\theta \in \mathbb{R}^m} \sum_{i=1}^{n} \left(\left(\sum_{j=1}^{m} \phi(x)_{ij} \theta_j \right) - y_i \right)^2. \end{aligned}$$

Checking for monotonicity on linearly interpolated lattice is relatively easy because the target function increases in a given direction only if the lattice parameters increase in that direction. Based on lattice regression, monotonic lattice regression [15] guarantees monotonicity between input feature and the output prediction by adding pairwise linear inequality constraints expressed as $A\theta \leq 0$ in the objective function of lattice regression:

$$\theta = \arg\min_{\theta \in \mathbb{R}^m} \sum_{i=1}^{n} \left(\hat{y}_i - y_i\right)^2, s.t.\ A\theta^T \leq 0. \quad (4)$$

The constraint matrix A is a m-column sparse matrix with one 1 and one -1 per row. m is the number of lattice nodes, and the row number of A is the total count of adjacent node pairs in the lattice structure. For example, if we want to constrain the output of the lattice in Figure 3(b) increasing with both of its two-dimensional features, we need to make sure that the parameters of lattice nodes along each dimension are monotonically increasing. To this end, we can add the inequality constraints as shown in Figure 4. Each row of A constrains the inequality relationship between each pair of adjacent lattice parameters, making it possible whether or not to constrain one specific feature. For example, the first row of matrix A gives the inequality relationship between θ_1 and θ_2 (i.e., $\theta_1 \leq \theta_2$). With the monotonic constraints, each feature can be independently left unconstrained, or constrained to be either monotonically increasing or decreasing.

4.2 Cell-Wise Regularizer

We add a novel cell-wise regularizer to Equation 4 that penalizes the parameter differences of pairs of adjacent lattice nodes with different weights, which controls the smoothness of the lattice model and makes the target function fit for the potential non-uniform CDF better.

Our key insight is that the cells in a lattice model should have different smoothness because of the non-uniform data distribution. The smoothness can be measured by the difference of lattice parameters between nodes of the same cell, for example, the differences of parameters between lattice nodes are small if the joint CDF of the covered area changes slightly, and such area is more smooth. Because the position of every pair of connected nodes in a cell differs only in one dimension, we can reduce the compatibility from regularizing nodes in one cell to regularizing each pair of adjacent nodes in the lattice structure. By doing so, the dimensionality of the joint CDF is also reduced to the CDF of every single dimension. In order to approximate the cumulative distribution of each dimension, we utilize pre-build equal-depth histograms on every single attribute. Histogram is an approximate representation of the probability distribution and the bin positions of equal-depth histograms are able to represent the data skew. The histogram can be built on randomly sampled data or retrieved from statistics information of database systems. Based on the approximated distribution of each dimension from the equal-depth histograms, we can assign different regularization weights to the squared-difference of each pair of adjacent lattice nodes by:

$$R(\theta) = \sum_{i=1}^{d} \sum_{\substack{C_r, C_s \text{ such that} \\ C_r \text{ and } C_s \text{ adjacent on dimension } i}} H_i(C_r, C_s)(\theta_r - \theta_s)^2,$$

where $H_i(C_r, C_s)$ is the cell-wise weight of lattice nodes C_r and C_s on dimension i, and we use the reciprocal of approximated distribution by the histogram as the cell-wise weight.

Different from previous regularizer that penalize lattice parameters with uniform constraints [15], [19], i.e. $H_i(C_r, C_s) = 1$, we can impose fine-grained controls on each lattice cell with this cell-wise regularizer. And controlling the smoothness of the lattice model helps it fit the target function better and avoid the risk of over-fitting. With the cell-wise regularizer, the objective function of our CDF learner becomes:

$$\theta = \arg\min_{\theta \in \mathbb{R}^m} \sum_{i=1}^{n} \left(\hat{y}_i - y_i\right)^2 + \lambda R(\theta), s.t.\ A\theta^T \leq 0, \quad (5)$$

where λ is the tuning parameter that decides how much we want to penalize the smoothness of the model.

4.3 Attribute-Aware Calibration

To achieve a good performance of accuracy with the lattice regression model, we will have to partition the space D into fine-grained lattices. That will bring more lattice nodes (thus more model parameters), which is unacceptable and contrary to our purpose of designing a lightweight model. In this paper, we resort to feature calibration [15], [20] to further improve the model accuracy while keeping a smaller model size.

To maintain the monotonicity, we adopt monotonic piece-wise linear (PWL) functions to calibrate each attribute of the sample x before it is fed to the lattice regression model. But different from former work [20] that sets the breakpoints uniformly spaced between 0 and $|dom(A_i)|$ for the PWL function of the i^{th} attribute A_i, we set the breakpoints in an attribute-aware method by putting more breakpoints in regions where data is dense. And in order to capture the data distribution of each attribute, we again utilize the pre-build equal-depth histograms.

Given a sample point x=$[x_1, x_2, \ldots, x_d]$ of space D, let C^i be the lattce size of attribute A_i (i.e., A_i is partitioned into $C^i - 1$ intervals with C^i lattice nodes), the i^{th} calibration funciton $g^i(x_i, \alpha^i)$ with J breakpoints $[b_1^i, b_2^i, \ldots, b_J^i]$ is defined as a PWL function with parameter α^i=$[\alpha_1^i, \alpha_2^i, \ldots, \alpha_J^i]$, where $0 \leq \alpha_j^i \leq C^i - 1$ is the function value of the j^{th} breakpoint of g^i and α^i is monotonically increasing with b^i.

Please note that the minimum value (α_1^i) and maximum value (α_J^i) of the PWL function map to the lattice bounds 0 and $C^i - 1$ of A_i. For example, the PWL function in Figure 3(b) shows the calibration for the first attribute ($i = 1$) of Forest dataset. Here $C^1 = 3$ and $J = 5$, i.e., the number of lattice nodes is 3, the interval number is 2 and the PWL function has 5 breakpoints. Position of the first breakpoint b_1^1 is 0 and the last b_5^1 equals to the domain of the current attribute. The minimum function value of b_1^1 is 0 (i.e., α_1^1=0), and likewise, the maximum function value of b_5^1 is 2 (i.e., α_5^1=2). Our attribute-aware calibration defines the positions of the other three internal breakpoints according to the histogram of the attribute. In this example, the distance between b_2^1 and b_3^1 is smaller than other intervals, indicating the data distribution is denser in this area and change of the cumulative distribution is larger here. The remaining 3 function values ($\alpha_2^1, \alpha_3^1, \alpha_4^1$) are unknown parameters and need to be learned from the data.

Let $g(x, \alpha)$ denote the vector function with i^{th} component calibration function $g^i(x_i, \alpha^i)$, and $g(x, \alpha)$ maps a sample point x to the domain $(C^1-1)\times(C^2-1)\times\ldots\times(C^d-1)$ of

the lattice. Let e_i denote the standard unit basis vector that is one for the i^{th} component and zero elsewhere with length d, then we can write:

$$g(x,\alpha) = \left(\sum_{i=1}^{d} \left(e_i^T g^i(e_i x^T, \alpha^i) \right) \right)^T . \tag{6}$$

We jointly optimize the calibration parameters and the lattice parameters to learn the calibrated lattice model like [15] with the following objective function:

$$\begin{aligned} \theta, \alpha = \arg\min_{\theta,\alpha} \sum_{i=1}^{n} \left(\left(\sum_{j=1}^{m} \phi(g(x,\alpha))_{ij} \theta_j \right) - y_i \right)^2 + \lambda R(\theta) \\ s.t.\ \ A\theta^T \leq 0 \ and \ B\alpha^T \leq 0 \ , \end{aligned} \tag{7}$$

where matrix B is used to set up monotonicity constraints for the PWL functions similar to A.

In our CDF learner, we use another monotonic PWL function as the output calibration layer. This PWL function takes in the raw output of monotonic lattice regression model and generates the learned CDF value. Since the valid values of cumulative distribution functions should lie in range $[0, 1]$, we constrain the minimum and the maximum value of the PWL function as zero and one to make the estimation of the model reasonable. Different from the attribute-aware calibration function, breakpoints of the last PWL function are uniformly distributed along the output of the lattice model.

Figure 3(b) shows the model structure of our CDF learner. There is first an attribute-aware calibration for each input range feature or an encoding operation for categorical predicates, then a lattice regression model taking all above features as input and generate intermediate results that guarantee monotonicity. Finally, there is an output calibration to constrain the estimation result between 0 and 1.

Before training the CDF learner, by setting values of A and B, we inject the pre-knowledge that the CDF value should be monotonic increasing with every numeric attribute to the lattice regression model. During training, we optimize the values of the breakpoints together with the lattice parameters using batched stochastic gradients.

4.4 MI-Based Ensemble

Although the lattice regression model has fewer trainable parameter number (see Section 6.5 for detailed analysis), the memory usage and computation complexity still grows exponentially with respect to the number of attributes. To improve the efficiency of the CDF learner when dealing with relatively large dimensional data, we adopt an ensemble method and propose a mutual information based attributes partition algorithm.

4.4.1 Lattice Ensemble

The lattice regression model provides accurate, reliable, and efficient selectivity estimation if there are a few range predicates. However, the model is computationally challenging to learn when facing relatively large dimensional queries. We adopt the idea of model ensemble [21], [22] by splitting the attributes into several subsets and training an ensemble of lattice regression models over these subsets. For example, Figure 5 shows an ensemble with five lattice models, and every model learns a subset of the total attributes.

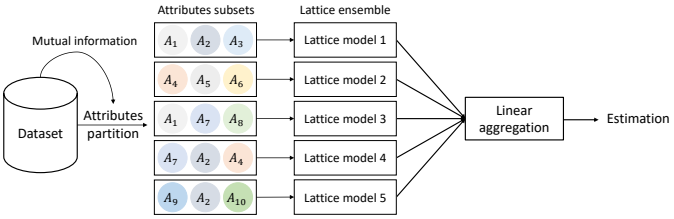


Fig. 5: Lattice Ensemble

Suppose the attributes are partitioned into L subsets (i.e. there are L lattice models in the ensemble) and each subset of attributes is $S[l]$, the estimation function is:

$$\hat{y} = F_X(x) = \sum_{l=1}^{L} a_l F_{X_{S[l]}}(x_{S[l]}), \tag{8}$$

where $F_{X_{S[l]}}$ is a monotonic lattice regression model on subset l, $x_{S[l]}$ are training data of the corresponding subset, and a_l is the weight of each model. In order to maintain the monotonicity of the ensemble, we constrain the weights to be positive, i.e., $a_l \geq 0$ for all l. It has been proved that the lattice ensemble is equivalent to a single lattice defined on all D attributes [22], so the accuracy can be guaranteed. And the ensemble method trains several small lattice model separately and parallelly, avoiding the high computation cost and large parameter size of a single lattice model.

4.4.2 MI-based Attributes Partition

Attributes partition affects the performance of ensemble [23], [24]. Previous ensemble methods partition attributes into different subsets randomly [25], according to the diversity of the attributes [24], or based on pre-computed linear interactions between pairwise attributes [22].

When learning a multi-dimensional CDF, a key challenge affecting the accuracy derives from the correlation between different dimension of the dataset. Our insight is that in order to learn the nonlinear correlation between the attributes, the highly-correlated attributes should be partitioned into one subset. There are several metrics measuring correlation between attributes and we design a mutual information based partition algorithm to determine which attributes should be combined in the same lattice.

Mutual information (MI) [26], [27] uses the concept entropy [26] to represent the mutual dependence between two random variables. The formal definition of MI of variables X and Y is given by

$$MI(X;Y) = \sum_{x \in X} \sum_{y \in Y} P_{XY}(x,y) log \frac{P_{XY}(x,y)}{P_X(x) P_Y(y)}, \tag{9}$$

where $P_{XY}(x,y)$ is the joint distribution of X and Y.

Mutual information not only measures the non-linear relationship of the joint probability distribution of attributes, but also is faster to compute compared to pairwise linear interactions derived from pre-computed models [22]. What's more, we can also use sampled data for MI calculation to accelerate the process. With mutual information of each pair of the attributes, we design a greedy algorithm for attributes

partition. The basic idea is to partition attributes with larger MI values into the same subset, meanwhile, all attributes must appear at once and attributes are also allowed to recur in different subsets.

Algorithm 1: MI-based Attributes Partition

input : X_d is the d-dimensional data,
L is the number of subsets,
S_l is the number of attributes in subset l,
A_i is the ith attribute ($i \in [1, d]$),
output: P is the partition result

1 $Pairs = \emptyset$ // added attribute pairs
2 $P_l = \emptyset$ for $l \in L$ // empty partition result
3 **for** *pairwise* i, j *in* d **do**
4 $MI_{i,j} = \texttt{MutualInformation}(A_i, A_j)$
5 **while** $|Pairs| < D$ *or* $\exists l, |P_l| < S_l$ **do**
6 **if** $|P_l| = S_l, \forall l \in L$ **then**
7 **for** A_i *not in* $|Pairs|$ **do**
8 $A_p, A_q = \texttt{MinIdx}(Pairs)$
9 **for** $P_l, l = [L, L-1, ..., 1]$ **do**
10 **if** $A_p \in P_l$ **then**
11 $\texttt{Remove}(P_l, A_p)$
12 $P_l \leftarrow A_p$
13 break
14 **if** $A_q \in P_l$ **then**
15 $\texttt{Remove}(P_l, A_q)$
16 $P_l \leftarrow A_q$
17 break
18 **else**
19 $A_p, A_q = \texttt{MaxIdx}(MI)$
20 **for** $P_l \in P$ **do**
21 **if** $A_p \in P_l$ & $A_q \in P_l$ **then**
22 $Pairs \leftarrow (A_p, A_q)$
23 break
24 **if** $A_p \in P_l$ & $|P_l| < S_l$ **then**
25 $P_l \leftarrow A_q,\ Pairs \leftarrow (A_p, A_q)$
26 break
27 **if** $A_q \in P_l$ & $|P_l| < S_l$ **then**
28 $P_l \leftarrow A_p,\ Pairs \leftarrow (A_p, A_q)$
29 break
30 **if** $S_l - |P_l| >= 2$ **then**
31 $P_l \leftarrow A_p, A_q,\ Pairs \leftarrow (A_p, A_q)$
32 break
33 return P

Algorithm 1 shows our MI-based attributes partition algorithm. We first compute pairwise mutual information of all attributes (line 3-4), and then use the greedy algorithm to partition D-dimensional attributes into L subsets. The partition process continues until all attributes appear at least once and all subsets are full (line 5). In the process, if there is no empty space in the subsets (line 6), the already partitioned pair of attributes with minimum mutual information are removed from the result (line 7-17).And if there is still spare space (line 18), the pair of attributes that haven't been added to the partition result with largest MI are selected as candidate pair (line 19). Then four cases of the selected attributes and current partition situation are examined, and these two attributes are greedily added into the partition result. (line 20-35).

For simplicity, we set number of attributes in each subset as the same value S_l. Note that more lattices in the ensemble doesn't always bring improvement in estimation accuracy, because an appropriate L would be enough to capture the correlations between the attributes. We heuristically use $1.5 \times D$ as $\sum_{l=1}^{L} S_l$ for predicates with dimensions larger than 6. For example, for the 10-D Forest dataset, we use an ensemble of 5 lattice model with 3 features in each subset.

4.5 Categorical Predicates Extension

Categorical attributes (such as date or string) and categorical predicates (such as IN or equality) are common in database queries. Most existing query-based selectivity estimation methods focus only on numerical attributes and range predicates. They downplay how to deal with categorical attributes and predicates by making attribute independent assumptions [9] or mapping categorical predicates into integers and then treat them as range predicates [12]. We handle queries with such predicates in a unified framework. For attribute of date type, we extract the predicates and calculate the differences from the earliest day of that attribute. By the above transformation, the calculated difference of date type attribute can be handled the same as other numerical attributes and there also exists monotonicity between the date and the joint CDF value.

For string attributes, equality and IN predicates are more common and it's obvious that there is no monotonicity between the attributes and the corresponding CDF. To handle queries with string attributes, we first encode the strings with one-hot embedding. For equality predicates, the input feature is the encoded vector of the string and for IN predicates, the input feature is the result of bitwise OR operator on all encoded vectors of the strings in the predicates. After encoding, the string features are treated as part of the input feature with other range predicates to the multi-dimensional CDF learner, but we don't need to set any monotonic constrain on the string features and the lattice regression model will treat them as normal features.

4.6 Theoretical Justification and Analysis

As mentioned in Section 1, in addition to accuracy, there are several other theoretical requirements for selectivity estimation of range predicates: (1) Users may want the estimated selectivity to be consistent and interpretable in applications like data exploration. Since the actual selectivity is monotonically increasing with the query range, when a greater range is given, a larger or equal number of results is preferable, so the user is able to interpret the selectivity for better analysis. (2) A good estimation is supposed to be stable and has validity property. Here we justify how MOSE can guarantee these theoretical requirements.

Lemma 1. *The multi-dimensional CDF $F_X(x)$ is monotonically increasing with each dimension of attribute x and the selectivity $sel(p)$ of query p is monotonically increasing with the range of the predicate.*

Lemma 1 can be easily proved by Section 4.1 and Section 4.3. In order to justify validity of MOSE, we will prove that the adoption of CDF to selectivity can help identify invalid queries.

Theorem 2. *Given query predicates p with range $R = [l_1, u_1] \times [l_2, u_2] \times \cdots \times [l_d, u_d]$, if $\exists j, l_j > u_j$ and $l_i < u_i, \forall i \neq j$, then $Sel(p) = 0$*

Proof. According to Theorem 1, $F(x_2) \leq F(x_1)$ where $l_j \in x_1$, $u_j \in x_2$ and x_1 and x_2 differs only in dimension j. Then according to Equation 3, $Sel(p) \leq 0$. We then use a clip function to eliminate such negative results to promise the validity of selectivity estimation. □

Theorem 3. *Given predicates p_1, p_2 and p_3 with query range $R_1 = [l_1, u_1] \times \cdots \times [l_k, m_k) \times \cdots \times [l_d, u_d]$, $R_2 = [l_1, u_1] \times \cdots \times [m_k, u_k] \times \cdots \times [l_d, u_d]$, and $R_3 = [l_1, u_1] \times \ldots [l_k, u_k] \times \cdots \times [l_d, u_d]$, where $l_i < u_i$ and $l_k < m_k < u_k$, there is $Sel(p_3) = Sel(P_2) + Sel(P_1)$.*

Proof. According to Equation 3,

$$\begin{aligned} Sel(p_1) + Sel(p_2) &= \sum_{\forall x_i \in \{l_i - 1, m_i - 1\}} \left\{ \left(\prod_{i=1}^{d} s(i) \right) F_X(x) \right\} \\ &+ \sum_{\forall x_i \in \{m_i - 1, u_i\}} \left\{ \left(\prod_{i=1}^{d} s(i) \right) F_X(x) \right\} \\ &= \sum_{\forall x_i \in \{l_i - 1, u_i\}} \left\{ \left(\prod_{i=1}^{d} s(i) \right) F_X(x) \right\} \\ &= Sel(p_3) \end{aligned}$$

□

Lastly for stability, once MOSE finishes optimization, the parameters are fixed and obviously the predicted selectivity for the same query will never change.

5 Active Data Generator

The quality of training data has an enormous effect on the performance of machine learning models [28]. In query-based selectivity estimation, most methods generate training and testing queries randomly, leading to some unexpected problems. For example, randomly generated queries tend to have a small selectivity especially when the queries contain multiple attributes. And such queries may also make the models fail to find and fit areas where selectivity changes dramatically. In this section, we propose an innovative active learning algorithm to generate representative training data and decrease the cost of query execution for data collection.

5.1 Active Learning for Regression

Supervised learning demands a large set of training data, but retrieving labels is often an expensive and time-consuming process. Active learning was proposed aiming to select as few data as possible to learn a model with satisfying accuracy [29], [30]. When there are plenty of unlabeled data, active learning attempts to overcome the labeling problem by deciding which unlabeled instances to be labeled by an expert (e.g., a human annotator or a database engine in our scenario).

Active learning has been explored in database field, ADCP [31] is an active data collection platform for query execution cost prediction and plan regression prediction. However, unlike the random forest regression model used in ADCP, there is no uncertainty in our lattice regression model. And most active learning methods are pool-based [32], [33], [34], meaning there is a pool of unlabeled data. But in query-based selectivity estimation, range predicates can be treated as points in an infinite query space. So pool-based active learning strategies cease to be effective in selectivity estimation. Facing the above challenges, we propose a novel active learning algorithm with a weighted-sampling strategy that is deeply combined with the monotonic lattice regression model. Figure 3(c) shows the process of our active data generator.

5.2 Weighted lattice sampling

Since the training data space is infinite in selectivity estimation, we avoid generating a pool of unlabeled data by converting the problem of points selection in an infinite space into the problem of picking the lattice cells that are most valuable or necessary to optimize. The root cause of the unsatisfactory estimation accuracy is that there are not enough training points in the cells to optimize the lattice parameters. Such deficiency might be caused by inadequate data distribution, making few points in each cell. Another reason lies in that selectivity changes dramatically in the cells, and the areas covered by such cells are the interesting areas we need to pay more attention to. So our goal is to generate more points in the cells where the predictions of current points are not accurate.

We considered two factors when choosing cells for query generation. The first is how accurate the cell is and the second is how many points are in the cell. For a cell c, suppose there are already M_c points in the cell and k_c of the M_c points are in the $TopK$ points with the top largest error evaluated by the current model. We combine the above two factors and calculate the sampling weight of c by $\mathcal{P}_c$:

$$\mathcal{P}_c = \frac{1 + \omega k_c}{1 + M_c},$$

where ω is the error weight. $\mathcal{P}$ is able to assign a larger weight to cells that have more inaccurate points and cells that have insufficient points. With the sampling weight, we select the candidate cells and generate one point in each of the cells randomly as active data set. With the process described above, the active data generator is deeply integrated with the lattice model in our CDF learner.

Algorithm 2: Active Data Generator

input : X_L is the initial labeled data,
θ is model weight of CDF learner,
$\mathcal{T}$ is the cost threshold of data collection,
ϵ is the cost function to label a data instance,
B is the number of data selected in one batch,
$\mathcal{P}$ is a function to calculate cell sampling weight
output: X is the labeled training data

1 $X \leftarrow X_L$ // total training set
2 $t \leftarrow 0$ // initialize total cost
3 **while** $t < \mathcal{T}$ **do**
4 $\quad \theta \leftarrow \texttt{TrainModelWith}(X)$
5 $\quad Error \leftarrow \texttt{Evaluate}(\theta, X)$
6 $\quad P_{TopError} \leftarrow \texttt{Top}(X, Error, K)$
7 $\quad \mathcal{P}_c \leftarrow X, P_{TopError}$
8 $\quad cells \leftarrow \texttt{WeightedSampling}(\mathcal{P}_c, B)$
9 $\quad X_A \leftarrow \texttt{RandomPointGenerate}(cells)$
10 $\quad X_{AL} \leftarrow \texttt{ExecuteQuery}(X_A)$
11 $\quad t = t + \epsilon(X_A)$
12 $\quad X = X \cup X_{AL}$
13 return X

Algorithm 2 shows the algorithm of our active learning strategy. In the beginning, our active data generator randomly generates initial labeled data X_L like [9]. Then the model is trained with the initial dataset (Line 4), and we evaluate the current data with the learned model to get the error of each point (Line 5). We then sort the points by their evaluation error and find the top K points with the largest error (Line 6). Based on these points, the sampling weight $\mathcal{P}_c$ of each cell is calculated(Line 7), and a batch of B

cells are sampled with replacement according to the weight (Line 8). Then a point is randomly generated in each of the B cells (Line 9) and added to the training data set (Line 12) after its selectivity is retrieved (Line 10), meanwhile, the cost of executing the query is added to the total cost t (Line 11). The above process will continue until the total cost exceeds the threshold. Not that the active data generator can directly be adopted to cases using lattice ensemble, because the weighted sampling strategy is also suitable for the partitioned subsets of attributes. With the weighted lattice sampling strategy, the active data generator can find valuable and representative training data sufficiently, helping to improve the accuracy of our CDF learner.

6 Experiments

6.1 Experimental setup

Methods We compare our CDF learner with two scan-based approaches, three query-based and one data-based learned selectivity estimators :

(1) AVI: This method creates equal-depth histograms for each attribute and estimates selectivity with attribute value independence assumption (AVI).

(2) Sampling: This method executes query predicates on a uniform random sample of data to make estimations.

(3) Lightweight model (LWM) [9]: This method combines range predicates with three additional statistical features (AVI, MinSel, and EBO) to make estimation. We use the XGBoost model as the regression model as the paper suggests.

(4) NN: We build a neural network to predict the selectivity, and the estimation process is the same as MOSE. The neural network firstly learns the joint CDF and then estimates the selectivity using the *Selectivity Computer*. The neural network consists of three fully-connected layers with hidden units of 128, 64, and 1. We use ReLU as intermediate activation functions and add a Sigmoid function for the last layer to constrain the output between 0 and 1.

(5) Naru [10]: Naru is a data-based estimator, and it adopts the deep autoregressive model and factorizes the joint data distribution into conditional distributions using the product rule. We use the source code and parameter setting shared by the authors [35].

(6) QuickSel [12]: QuickSel is the-state-of-the-art query-based selectivity estimation method. It relies on a mixture model to capture the underlying distribution of the data. We use the source code shared by the authors [36].

Datasets We use two real-world datasets and one synthetic benchmark:

(1) DMV [37]: This dataset contains the vehicle registration records of New York State with 11,591,878 rows. We build joint CDF on the following numerical attributes: Model Year, Zip, and Unladen Weight.

(2) Forest [17]: The origin dataset contains 54 attributes and 581012 rows. Like [9] [38], we ignore the binary attributes and use the first 10 continuous numeric attributes.

(3) TPC-H [39]: We use the skewed TPC-H benchmark generator [40] in our experiments, and this generator has a parameter z that controls the degree of data skewness. When z = 0, it generates a uniform distribution, and as z increases, the data becomes more and more skewed. We create a 1GB highly-skewed dataset with z = 8 and we use different kinds of attributes including number, string, and date types from the Orders table. This table has 1.5 million rows totally.

Queries The range predicates of our training queries and testing queries are generated using the random-centric method like [9]. For the TPC-H dataset, we generate the categorical attributes randomly from its candidate domain. We generate 5000 range queries and execute them to get the selectivity. We then take 1000 of them as test set and the other 4000 as training set pool.

Environment We conduct all our experiments on a computer with a 3.00GHz Intel Core i5-8500 CPU and 24GB RAM running Ubuntu 20.04.

6.2 Selectivity Estimation

In this section, we compare the estimation accuracy of our monotonic selectivity estimator MOSE and other methods. To evaluate how the number of training data influences the learning-based methods, we increased the size of training data from 200 to 1000 by adding 100 queries each time. Note that for MOSE, the training data, except the randomly initialed 200 instances, are generated by the *Active data generator*. And for other learning-based methods, training data was randomly selected from the already generated training set pool. We set the lattice size of each attribute heuristically according to their domain range and the total number of dimensions, and we used 50 breakpoints in each piece-wise linear function in our method (see Section 6.6 for hyperparameter tunning). We run all our experiments five times to get statistically meaningful results. Like QuickSel, we use root mean square error (RMSE) as our evaluation metrics to measure the estimation error between the predicted selectivity $\hat{y}$ and the true selectivity y:

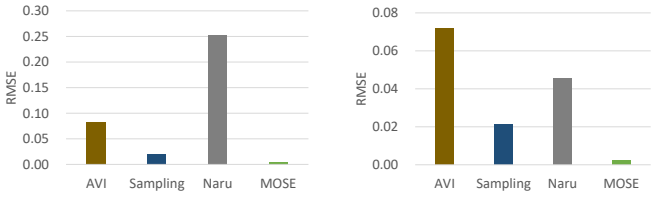
$$\mathrm{RMSE} = \left(\frac{1}{N} \sum_{i=1}^{N} (\hat{y} - y)^2 \right)^{1/2}.$$

Accuracy on DMV Data Set Table 2 demonstrates the estimation results on 3-dimensional DMV dataset. We show results with training data sizes of 200, 400, 600, 800, and 1000 to save space, but the whole result is in the same pattern. Given the same number of training queries on DMV dataset, MOSE outperforms all other learning-based methods and is far better especially when there are more training queries. We notice that the LWM is better than NN at first, but as the number of training queries increases, NN outperforms LWM. The state-of-the-art method QuickSel has the least error among the former learning-based methods, but the RMSE has a slowing rate of decline with an increasing number of queries. The reason is QuickSel relies on a uniform mixture model to fit the data distribution and it uses a sampling based method to define the hyperrectangles of the uniform distribution function in their model. Considering the non-uniform data distribution in DMV dataset, the observation of more queries forces the uniform distribution functions to care for more space and brings limited improvement. On the other hand, MOSE gives the best estimation thanks to the accurate CDF learner with specially extended monotonic lattice regression model. Compared to QuickSel, our estimator provides predictions with 59% to 68% less error with the same number of training data.

TABLE 2: Selectivity estimation accuracy on DMV

Estimator	Training data size				
	200	400	600	800	1000
LWM	0.03474	0.02576	0.01817	0.01758	0.01607
NN	0.04787	0.03082	0.02153	0.01803	0.01577
QuickSel	0.02151	0.01421	0.01296	0.01125	0.01027
MOSE	**0.00674**	**0.00543**	**0.00463**	**0.00429**	**0.00393**

Since Naru and the two traditional scan-based methods AVI and Sampling don't rely on training query size, we don't compare their results in Table 2. Figure 6a demonstrates the comparison between our method with Naru, AVI and Sampling on DMV dataset. For AVI method, we build equal-depth histograms on each attribute with 100 buckets from 30000 rows randomly sampled from data [41], and we use a sample of 100 tuples for Sample as QuickSel did. For Naru, we use 2000 as the number of progressive samples to use per query. Among all the methods, Naru has the worst estimation accuracy, and the reason is that Naru is better at point queries rather than range queries. What's more, the sampling method in Naru makes the estimated result unstable and increases the estimation cost. The accuracy of AVI is not satisfying mainly because data correlation exists between different attributes while AVI makes attribute independence assumptions. MOSE also outperforms Sampling, because the Model Year attribute in DMV is very skewed and Sampling fails to estimate accurately due to the non-uniform data distribution.



(a) RMSE on DMV (b) RMSE on TPC-H

Fig. 6: Comparison with scan-based methods

Accuracy on Forest Data Set To evaluate the accuracy of different models with respect to different numbers of attributes, we conduct experiments on 10-D Forest dataset. Table 3 shows the estimation accuracy of different learning-based methods when the attribute number is increased from 2 to 10. Due to space limitations, we here mainly demonstrate the estimation result with 1000 training queries on even number of dimension attributes and all the query-based methods are trained with 1000 queries. It can be found that our estimator MOSE outperforms all the other learned models and scan-based methods, and MOSE produces up to 23% more accurate estimations than QuickSel when the range query consists of 6 attributes. Note that the estimation error is dropping with more dimension of predicates, and the reason is that the selectivity of the range query with high dimensional attributes tends to be small because of the attribute correlation, and this is unavoidable to some extent.

Accuracy on Hybrid TPC-H Data Set We conduct experiments on the TPC-H dataset to evaluate our selectivity estimator when there exist categorical predicates in queries. Since the other query-based learning method downplayed handling categorical predicates, we mainly compare with Naru and the scan-based methods. When dealing with categorical attributes, database systems such as PostgreSQL use most common value (MCV) as statistics additional to histograms. We also build the MCV statistics on the string attribute in the AVI method so as to make it fair.

TABLE 3: Selectivity estimation accuracy on Forest

Estimator	Range predicates dimension				
	2D	4D	6D	8D	10D
AVI	0.23020	0.06069	0.01060	0.00240	0.000582
Sampling	0.00642	0.01164	0.00452	0.00946	0.000718
Naru	0.20113	0.59320	0.56103	0.10131	0.308497
LWM	0.03125	0.01573	0.00729	0.00229	0.000574
NN	0.00638	0.01226	0.00943	0.00240	0.000582
QuickSel	0.00470	0.00773	0.00382	0.83949	0.000590
MOSE	**0.00419**	**0.00544**	**0.00274**	**0.00223**	**0.000555**

Result in Figure 6b shows that the accuracy of our estimator far exceed Naru and AVI, and gains about one-eighth RMSE of Sampling on hybrid predicates. Even though there is MCV for AVI method, it fails because of the correlation in the attributes. And the generated table follows Zipf distribution and is highly skewed, making Sampling unable to estimate accurately on the sampled tuples. The sampling method for range predicates in Naru leads to poor performance as in other datasets. Our estimator, however, produces accuracy selectivity estimation no matter there are numerical or categorical predicates.

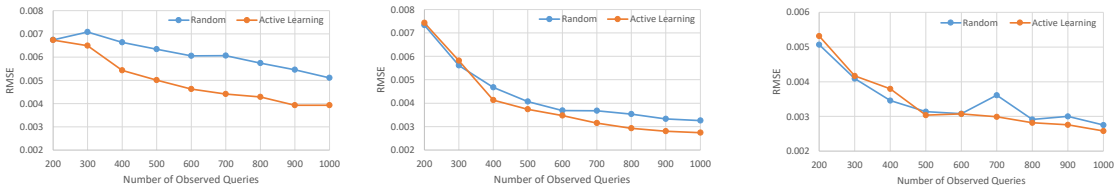
6.3 Ablation Experiments

In this section, we demonstrate the effectiveness of the proposed cell-wise regularizer, attribute-aware calibration and lattice ensemble.

TABLE 4: Combination of calibration and regularizer

Combination method	RMSE
Laplacian regularizer + Uniform calibration	0.00713
Laplacian regularizer + A-A calibration	0.00540
C-W regularizer + Uniform calibration	0.00530
C-W regularizer + A-A calibration	**0.00393**

Effectiveness of Regularizer and Calibration This set of experiments are used to test and verify the effectiveness of the cell-wise (C-W) regularizer, attribute-aware (A-A) calibration and lattice ensemble. Table 4 shows different combinations of regularizers and calibration methods on DMV data. We compared the C-W regularizer with the Laplacian regularizer [15]. For A-A calibration, we use uniform calibration as a baseline. From the table, the combination of Laplacian regularizer and uniform calibration has the largest error. The Laplacian regularizer penalizes adjacent lattice nodes in a cell with equal weights, which



(a) Active learning on DMV (b) Active learning on Forest (c) Active learning on TPC-H

Fig. 7: Active data generation

ignores the different distribution of each dimension. And uniform calibration sets breakpoints uniformly, limiting the adaptability of the PWL functions. The result shows both C-W regularizer and A-A calibration can improve the estimation accuracy thanks to the additional information about the data distribution. And MOSE outperforms all the other combinations, proving the effectiveness of our cell-wise regularizer and attribute-aware calibration.

TABLE 5: Efficiency Improvement of Lattice Ensemble

Forest 10D	Single Lattice	Lattice Ensemble
Number of Parameters	59599	**1140**
Training Time (ms/query)	996.030	**18.650**
Testing Time (ms/query)	184.785	**11.462**
RMSE	0.002472	**0.000555**

Effectiveness of Lattice Ensemble Table 5 demonstrates the performance comparison of single lattice model and lattice ensemble on 10D Forest dataset. It's obvious that the ensemble method dramatically decrease the number of parameters and save the cost of training and testing. Besides, lattice ensemble also improves the estimation accuracy, because for a single 10 dimensional lattice model, the training data is very sparse. But the ensemble method makes training samples more dense in each cell of the ensemble. The result indicates MOSE benefits from the MI-based ensemble.

6.4 Active Data Generating Impact

This set of experiments are used to test how the lattice-based active data generating strategy influences the estimation accuracy. We use uniform random sampling as the data generation baseline method and compare our active generator with it. We took 200 randomly generated instances as initial training data and then actively or randomly selected 100 samples as one batch of new data to update the training dataset. We recorded the RMSE of each batch and Figure 7a-7c demonstrate the comparison of the two different data augment strategies.

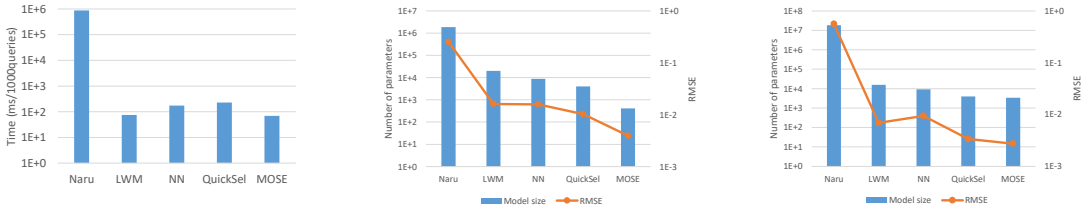
The orange lines in the figures are the downtrends of RMSE using our active data generator and the blue lines are test RMSE with random sampling approach. It's obvious that the orange lines are almost always lower than the blue ones, meaning that our active learning strategy for training data is effective.

It can be found that the RMSE almost always decreases when new batches of data are added to training data with our active learning strategy. However, the RMSE of random uniform sampling doesn't decrease continuously with new data, for example in Figure 7c, the test error swings up and down with new batches of data. The instability of uniform sampling strategy illustrates the randomly generated data is not always valuable for the model and might even be adverse to the performance of the estimator. The results suggest that active data generator produces more valuable instances than random sampling no matter when there are multiple dimensional predicates or predicates with categorical data types. And our active data generator produces a similar estimation as random generation with up to 50% less training data. For example, in Figure 7a, the RMSE achieves under 0.005 with 500 queries generated by our active data generator, but the randomly generated training data has a larger error even with 1000 samples. What's more, in all the experiments, our active data generator produces a lower RMSE that the random strategy can't achieve, meaning that the lattice-based active learning strategy helps to find candidate queries that are more representative, and the total generated data is in high quality compared with the randomly selected data.

6.5 Model Efficiency and Size

We compare the estimation time of learning-based methods. Figure 8a shows the time of different methods to produce one thousand estimates on DMV dataset. The data-based learned estimator Naru has the largest prediction cost because Naru relies on sampling to estimate selectivity for range queries. And although the XGBoost model used in LWM is very fast, it is slowed down because the LWM method needs to collect additional statistical information during estimation. However, MOSE only considers such information during the offline training process and uses pure query features when making estimations. And although the NN method has the same process as our CDF learner, its estimation cost is larger because the lattice regression model we use is well designed for fast evaluation. MOSE also outperforms QuickSel because QuickSel needs to calculate the overlap of one query hyperrectangle with 1000 hyperrectangles (i.e., the fitted multi-dimensional uniform distributions). In summary, MOSE provides accurate results with a lightweight model and fast estimation, making it suitable for application in database systems.

We also compare the model size of different learning-based methods. Like [9], we do not consider AVI and Sampling here because histograms are already in systems and might be useful for other purposes and sampling actually is a part of materialized table without any parameters. We follow [9] and [35] to set the hyperparameters of



(a) Model efficiencies (b) Models on DMV (c) Models on Forest

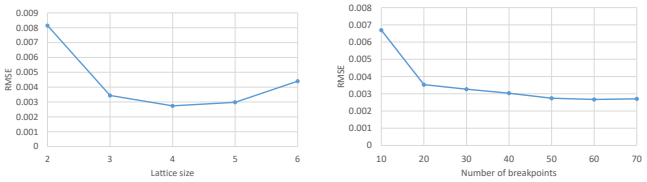
Fig. 8: Model efficiency and size

XGBoost model in LWM and the MADE model in Naru. Then we save the models and count the number of the trainable parameters to compare with other methods. For a d-dimensional range predicate, the size of the NN model we used in our experiments is $128 * d + 8449$, depending on both the hidden units in each layer and the dimension of the input features. The parameter number of QuickSel was set as $Min(4 * training\_size, 4000)$ as the author suggests. In our CDF learner, the total number of parameters is $m + \sum_{i=1}^{d} J_i + M$, where m is the number of lattice nodes, J_i is the number of breakpoints in the PWL function of ith attribute and M is the number of breakpoints in the output calibration function.

Figure 8b shows the model size and accuracy results on the DMV dataset and all these models are trained with 1000 queries. Our estimator has the smallest number of parameters and produces the most accurate results among all these methods. Compared to QuickSel, we achieved 61.8% less error with about one-tenth parameters of QuickSel. Figure 8c demonstrates the results on the 6D Forest dataset, we also have the best accuracy while keeping the smallest model size.

6.6 Hyperparameters Impact

To optimize our model's performance, we evaluate our method with different hyperparameters. In particular, we focus on the influence of lattice size and the number of breakpoints in attribute-aware calibration functions. For simplicity, we set the lattice size of each attribute the same and use the same number of breakpoints in both the input calibration layer and the output layer here, but more subtly hyperparameter selection can be conducted according to the domain range of each attribute.



(a) Lattice size vs. RMSE (b) Breakpoints vs. RMSE

Fig. 9: Hyperparameters Impact

We first constrain the breakpoints of the PWL functions to 50 and increase the lattice size. Figure 9a shows the influence of lattice size on Forest. The error is large when the lattice size is two because there is only one cell covering the whole query range. The RMSE drops drastically when lattice size is larger than 2, but then slightly changes until it starts to increase when it is larger than 5. The reason why the accuracy is not improving anymore is that a moderate lattice size is enough to split the range space into fine-grained cells, but the large lattice size makes the training data sparse in each cell and reduces the model accuracy. What's more, a larger number of lattice size also increases the model training time, so we use lattice size no more than 6 in our experiments and we set the same lattice size for each attribute for simplicity. Figure 9b demonstrates RMSE under different number of breakpoints when the lattice size is 4. The relationship between accuracy and the number of breakpoints is also very clear. More breakpoints don't always bring higher accuracy because there are not enough points in each interval of the PWL functions and they can't fit the features well. To achieve higher accuracy while keeping a small model size, we set the number of breakpoints in every PWL function to 50, and all the above experiments prove such settings are very effective.

7 Related Work

Selectivity estimation attracts much attention because of its importance to query optimization. Traditional selectivity estimation methods include histograms [3], [4], sampling [5], [6], and kernel-based approaches [38], [42], we refer readers to [43] for a detailed survey. Here we mainly offer a brief review of studies on selectivity estimation with machine learning models.

Data-based Selectivity Estimators These methods capture the joint probability distribution directly from data [10], [11], [44], [45], [46], [47]. Traditional histograms are actually data-based but they make attribute value independence assumptions for joint data distribution. Learning-based methods, on the contrary, try to avoid such assumptions and treat the estimation task as an unsupervised learning problem. DeepDB [11] adopts the deep probabilistic model Sum-Product-Network to capture the joint probability distribution of a given data set. But DeepDB doesn't fully drop the local independence assumption and the strongly correlated attributes might cause the model to be extremely large [14]. Bayesian networks are used in [46], [47] to model the inherent conditional independencies between attributes. Such approaches don't rely on independence assumptions anymore. However, learning the Bayesian network structure from data is an NP-hard problem [48], [49]. Another notable kind of data-based selectivity estimators such as Naru [10], MADE [45], and NeuroCard [44] utilize deep autoregressive

model to approximate the conditional probability distributions between attributes. However, such methods produce estimation for range queries slowly because they have to compute the selectivity based on sampling. What's more, it might take dozens of minutes to train the autoregressive model, making them unreliable in database systems.

Query-based Selectivity Estimators These methods build feedback between query results and queries to train estimation models. STHole [3] and ISOMER [50] are representative query-driven histograms that continuously adjust buckets number and boundaries according to the actual selectivity retrieved after executing the queries. But the number of buckets grows exponentially as the number of training queries increases. Selectivity estimation can be also treated as a supervised regression problem [9], [12], [13], [51], [52]. These methods extract range predicates as feature vectors and take the retrieved selectivity after executing the query as labels. Then different machine learning models for regression tasks can be adopted to deal with the estimation problem without any assumptions. For example, [9] adopts gradient boosted trees to predict the selectivity based on query features and three statistical features generated from histograms. MSCN [13] also uses query features along with additional information to train a multi-set convolutional network for selectivity estimation. They materialize table samples and execute range queries to get the positions of the qualifying samples, then the bitmaps of the positions are taken as part of the feature vectors. QuickSel [12] learns data distribution with a multi-dimensional uniform mix model, and it only uses range features to train the model but provides quite accurate results. However, our experiments show that the uniform mixture model can't produce satisfying estimation especially when the data distribution is highly skewed. The CDF learner we proposed in this paper estimates selectivity in a query-based fashion, and we produce estimation with better interpretability and higher accuracy than former works.

8 Conclusion

In this paper, we propose a monotonic selectivity estimator MOSE. We adopt lattice regression model to learn the joint cumulative distribution function and then estimate the selectivity. We propose an effective cell-wise regularizer and an attribute-aware calibration method to adapt the lattice model to CDF learning task. And we design a mutual information based attributes partition algorithm for lattice ensemble to improve the efficiency of MOSE. We also extend the MOSE for categorical predicates. We design an active learning strategy to retrieve valuable and representative training data. Experiments on both real-world and synthetic datasets show that MOSE outperforms existing learning-based selectivity estimation methods.

Acknowledgments

This work was partially supported by the National Key Research and Develop Plan under Grant 2018YFB1004401, National Natural Science Foundation of China under Grant 61772537, 61772536, 62072460, 62076245, and 62172424, Beijing Natural Science Foundation under Grant 4212022, and the Research Funds of Renmin University of China under Grant 20XHN126.

References

[1] V. Leis, A. Gubichev, A. Mirchev, P. A. Boncz, A. Kemper, and T. Neumann, "How good are query optimizers, really?" *Proc. VLDB Endow.*, vol. 9, no. 3, pp. 204–215, 2015.

[2] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, "Access path selection in a relational database management system," in *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, USA, May 30 - June 1*. ACM, 1979, pp. 23–34.

[3] N. Bruno, S. Chaudhuri, and L. Gravano, "Stholes: A multidimensional workload-aware histogram," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001*. ACM, 2001, pp. 211–222.

[4] Y. E. Ioannidis, "The history of histograms (abridged)," in *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003*. Morgan Kaufmann, 2003, pp. 19–30.

[5] Y.-L. Wu, D. Agrawal, and A. El Abbadi, "Applying the golden rule of sampling for query estimation," in *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 449–460.

[6] W. Wu, J. F. Naughton, and H. Singh, "Sampling-based query re-optimization," in *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*. ACM, 2016, pp. 1721–1736.

[7] A. Pavlo, G. Angulo, J. Arulraj, H. Lin, J. Lin, L. Ma, P. Menon, T. C. Mowry, M. Perron, I. Quah, S. Santurkar, A. Tomasic, S. Toor, D. V. Aken, Z. Wang, Y. Wu, R. Xian, and T. Zhang, "Self-driving database management systems," in *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chaminade, CA, USA, January 8-11, 2017*.

[8] T. Kraska, M. Alizadeh, A. Beutel, E. H. Chi, A. Kristo, G. Leclerc, S. Madden, H. Mao, and V. Nathan, "Sagedb: A learned database system," in *CIDR 2019, 9th Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 13-16, 2019*, 2019.

[9] A. Dutt, C. Wang, A. Nazi, S. Kandula, V. R. Narasayya, and S. Chaudhuri, "Selectivity estimation for range predicates using lightweight models," *Proc. VLDB Endow.*, vol. 12, no. 9, pp. 1044–1057, 2019.

[10] Z. Yang, E. Liang, A. Kamsetty, C. Wu, Y. Duan, P. Chen, P. Abbeel, J. M. Hellerstein, S. Krishnan, and I. Stoica, "Deep unsupervised cardinality estimation," *Proc. VLDB Endow.*, vol. 13, no. 3, pp. 279–292, 2019.

[11] B. Hilprecht, A. Schmidt, M. Kulessa, A. Molina, K. Kersting, and C. Binnig, "Deepdb: Learn from data, not from queries!" *Proc. VLDB Endow.*, vol. 13, no. 7, pp. 992–1005, 2020.

[12] Y. Park, S. Zhong, and B. Mozafari, "Quicksel: Quick selectivity learning with mixture models," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 2020, pp. 1017–1033.

[13] A. Kipf, T. Kipf, B. Radke, V. Leis, P. A. Boncz, and A. Kemper, "Learned cardinalities: Estimating correlated joins with deep learning," in *CIDR 2019, 9th Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 13-16, 2019*, 2019.

[14] X. Wang, C. Qu, W. Wu, J. Wang, and Q. Zhou, "Are we ready for learned cardinality estimation?" *Proc. VLDB Endow.*, vol. 14, no. 9, pp. 1640–1654, 2021. [Online]. Available: http://www.vldb.org/pvldb/vol14/p1640-wang.pdf

[15] M. R. Gupta, A. Cotter, J. Pfeifer, K. Voevodski, K. R. Canini, A. Mangylov, W. Moczydlowski, and A. V. Esbroeck, "Monotonic calibrated interpolated look-up tables," *J. Mach. Learn. Res.*, vol. 17, pp. 109:1–109:47, 2016.

[16] C.-T. Ho, R. Agrawal, N. Megiddo, and R. Srikant, "Range queries in olap data cubes," in *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '97, New York, NY, USA, 1997, p. 73–88.

[17] https://archive.ics.uci.edu/ml/datasets/Covertype.

[18] A. Dutt, C. Wang, V. R. Narasayya, and S. Chaudhuri, "Efficiently approximating selectivity functions using low overhead regression models," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 2215–2228, 2020.

[19] E. K. Garcia, R. Arora, and M. R. Gupta, "Optimized regression for efficient function evaluation," *IEEE Trans. Image Process.*, vol. 21, no. 9, pp. 4128–4140, 2012.

[20] Y. Wang, C. Xiao, J. Qin, R. Mao, M. Onizuka, W. Wang, and R. Zhang, "Consistent and flexible selectivity estimation for high-dimensional data," *CoRR*, vol. abs/2005.09908, 2020.

[21] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2004.

[22] M. M. Fard, K. Canini, A. Cotter, J. Pfeifer, and M. Gupta, "Fast and flexible monotonic functions with ensembles of lattices," in *NIPS*, 2016.

[23] Y. Ye, Q. Wu, J. Huang, M. Ng, and X. Li, "Stratified sampling for feature subspace selection in random forests for high dimensional data," *Pattern Recognit.*, vol. 46, pp. 769–787, 2013.

[24] L. Zhang and P. Suganthan, "Random forests with ensemble of feature spaces," *Pattern Recognit.*, vol. 47, pp. 3429–3437, 2014.

[25] T. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 832–844, 1998.

[26] C. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, 1948.

[27] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.

[28] V. Sessions and M. Valtorta, "The effects of data quality on machine learning algorithms," in *Proceedings of the 11th International Conference on Information Quality, MIT, Cambridge, MA, USA, November 10-12, 2006*. MIT, 2006, pp. 485–498.

[29] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, 2001.

[30] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[31] L. Ma, B. Ding, S. Das, and A. Swaminathan, "Active learning for ML enhanced database systems," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 2020, pp. 175–191.

[32] J. O'Neill, S. J. Delany, and B. MacNamee, "Model-free and model-based active learning for regression," in *Advances in computational intelligence systems*. Springer, 2017, pp. 375–386.

[33] R. Burbidge, J. J. Rowland, and R. D. King, "Active learning for regression based on query by committee," in *International conference on intelligent data engineering and automated learning*. Springer, 2007, pp. 209–218.

[34] Y. Baram, R. El-Yaniv, and K. Luz, "Online choice of active learning algorithms," in *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*. AAAI Press, 2003, pp. 19–26.

[35] https://github.com/naru-project/naru.

[36] https://github.com/illinoisdata/quicksel.

[37] https://catalog.data.gov/dataset/vehicle-snowmobile-and-boat-registrations.

[38] M. Heimel, M. Kiefer, and V. Markl, "Self-tuning, gpu-accelerated kernel density models for multidimensional selectivity estimation," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. ACM, 2015, pp. 1477–1492.

[39] http://www.tpc.org/tpch/.

[40] https://www.microsoft.com/en-us/download/details.aspx?id=52430.

[41] S. Chaudhuri, R. Motwani, and V. R. Narasayya, "Random sampling for histogram construction: How much is enough?" in *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*. ACM Press, 1998, pp. 436–447.

[42] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi, "Selectivity estimators for multidimensional range queries over real attributes," *VLDB J.*, vol. 14, no. 2, pp. 137–154, 2005.

[43] G. Cormode, M. N. Garofalakis, P. J. Haas, and C. Jermaine, "Synopses for massive data: Samples, histograms, wavelets, sketches," *Found. Trends Databases*, vol. 4, no. 1-3, pp. 1–294, 2012.

[44] Z. Yang, A. Kamsetty, S. Luan, E. Liang, Y. Duan, X. Chen, and I. Stoica, "Neurocard: One cardinality estimator for all tables," vol. 14, no. 1, p. 61–73, 2020.

[45] S. Hasan, S. Thirumuruganathan, J. Augustine, N. Koudas, and G. Das, "Deep learning models for selectivity estimation of multi-attribute queries," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 2020, pp. 1035–1050.

[46] L. Getoor, B. Taskar, and D. Koller, "Selectivity estimation using probabilistic models," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001*. ACM, 2001, pp. 461–472.

[47] K. Tzoumas, A. Deshpande, and C. S. Jensen, "Lightweight graphical models for selectivity estimation without independence assumptions," *Proc. VLDB Endow.*, vol. 4, no. 11, pp. 852–863, 2011.

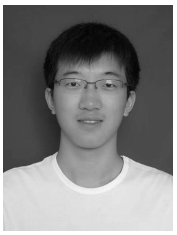
[48] R. Zhu, Z. Wu, Y. Han, K. Zeng, A. Pfadler, Z. Qian, J. Zhou, and B. Cui, "FLAT: fast, lightweight and accurate method for cardinality estimation," *Proc. VLDB Endow.*, vol. 14, no. 9, pp. 1489–1502, 2021.

[49] M. Scanagatta, A. Salmerón, and F. Stella, "A survey on bayesian network structure learning from data," *Prog. Artif. Intell.*, vol. 8, no. 4, pp. 425–439, 2019. [Online]. Available: https://doi.org/10.1007/s13748-019-00194-y

[50] U. Srivastava, P. J. Haas, V. Markl, M. Kutsch, and T. M. Tran, "ISOMER: consistent histogram construction using query feedback," in *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*. IEEE Computer Society, 2006, p. 39.

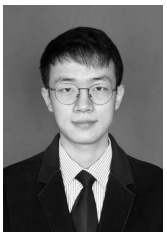
[51] T. Malik, R. C. Burns, and N. V. Chawla, "A black-box approach to query cardinality estimation," in *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007*, 2007, pp. 56–67.

[52] Y. Wang, C. Xiao, J. Qin, X. Cao, Y. Sun, W. Wang, and M. Onizuka, "Monotonic cardinality estimation of similarity selection: A deep learning approach," in *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 2020, pp. 1197–1212.



Luming Sun received the BSc degree from Xi'an Jiaotong University, China, in 2017. He is currently a PhD student at the School of Information and the Key Lab of Data Engineering and Knowledge Engineering, Renmin University of China. His research interests include query optimization and machine learning for systems.

Cuiping Li received the BE and ME degrees from the Xi'an Jiaotong University, China, in 1994 and 1997, respectively, and the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, China, in 2003. She is a professor with the School of Information, Renmin University of China. Her research interests include database systems, social networks, recommender systems, and big data analysis. She is a distinguished member of the CCF.



Tao Ji received the BSc degree from Xi'an Jiaotong University, China, in 2019. He is currently a PhD student at the School of Information and the Key Lab of Data Engineering and Knowledge Engineering, Renmin University of China. His research interests include query optimization and machine learning for systems.

Hong chen received the BE and ME degrees from the Renmin University of China, in 1988 and 1997, respectively, and the PhD degree from the Institute of Computing Technology, Chinese Academy of Sciences, China, in 2000. She is a professor with the School of Information, Renmin University of China. Her research interests include data privacy, big data management, and data analysis based on new hardwares. She is a distinguished member of the CCF.