Learning-based Property Estimation with Polynomials

JIAJUN LI*, Renmin University of China, China
RUNLIN LEI, Renmin University of China, China
SIBO WANG, The Chinese University of Hong Kong, China
ZHEWEI WEI†, Renmin University of China, China
BOLIN DING, Alibaba Group, China

The problem of estimating data properties using sampling frequency histograms has attracted extensive interest in the area of databases. The properties include the number of distinct values (NDV), entropy, and so on. In the field of databases, property estimation is fundamental to complex applications. For example, NDV estimation is the foundation of query optimization, and entropy estimation is the foundation of data compression. Among them, methods originating from statistics exhibit desirable theoretical guarantees but rely on specific assumptions about the distribution of data, resulting in poor performance in real-world applications. Learning-based methods, which use information from training data, are adaptable in the real world but often lack theoretical guarantees or explainability. In addition, a unified framework for estimating these frequency-based estimators with machine learning is lacking. Given the aforementioned challenges, it is natural to wonder if a unified framework with theoretical guarantees can be established for property estimation. The recent literature has presented theoretical studies that propose estimation frameworks based on polynomials. These studies also prove estimation errors with respect to the sample size. Motivated by the above polynomial estimation framework, we propose a learning-based estimation framework with polynomial approximation, which aims to learn the coefficients of the polynomial, providing theoretical guarantees to the learning framework. Through comprehensive experiments on both synthetic and real-world datasets for estimating various data properties like NDV, entropy, and power sum, our results show the superiority of our algorithms over previous estimators.

CCS Concepts: • **Computing methodologies** → *Learning linear models.*

Additional Key Words and Phrases: Property Estimation, Number of Distinct Values, Entropy, Polynomial, Learning

ACM Reference Format:
Jiajun Li, Runlin Lei, Sibo Wang, Zhewei Wei, and Bolin Ding. 2024. Learning-based Property Estimation with Polynomials. *Proc. ACM Manag. Data* 2, 3 (SIGMOD), Article 148 (June 2024), 27 pages. https://doi.org/10.1145/3654994

*Work partially done at the Chinese University of Hong Kong.

†Zhewei Wei is the corresponding author. The work was partially done at Gaoling School of Artificial Intelligence, Beijing Key Laboratory of Big Data Management and Analysis Methods, MOE Key Lab of Data Engineering and Knowledge Engineering, and Pazhou Laboratory (Huangpu), Guangzhou, Guangdong 510555, China.

Authors' addresses: Jiajun Li, Renmin University of China, China, 2015201613@ruc.edu.cn; Runlin Lei, Renmin University of China, China, runlin_lei@ruc.edu.cn; Sibo Wang, The Chinese University of Hong Kong, China, swang@se.cuhk.edu.hk; Zhewei Wei, Renmin University of China, China, zhewei@ruc.edu.cn; Bolin Ding, Alibaba Group, China, bolin.ding@alibaba-inc.com.

ACM 2836-6573/2024/6-ART148
https://doi.org/10.1145/3654994

1 INTRODUCTION

Estimating the properties of a discrete distribution from its samples is a fundamental problem in various fields. For example, estimating the number of distinct values (NDV), which is one of the most representative properties in the database, helps improve the query efficiency in database management [14]. Specifically, Table 4 in [42] demonstrates that improvements in cardinality estimation, rooted in precise NDV assessments, can lead to a reduction in query execution time by up to 50%. Additionally, systems like Spark and PostgreSQL (PG) directly relies on NDV to calculate cardinality [5, 6]. Similarly, applications in Biology for estimating unseen species [16, 51], Network studies for estimating virtualized network device cardinality [24], Linguistics for entropy evaluation [11], and Statistics for discrete distribution support size assessment [38] continue to emerge. Besides estimating NDV and entropy for the database, estimating properties like power sum [35], normalized support coverage [7], and distance to uniformity [39] is also critical when applied to practical scenarios including streaming and machine learning, demonstrating the importance of property estimation.

To estimate property, given that these properties in discrete distributions can be deduced from population frequency or the profile, samples are typically used to estimate the maximum likelihood of the actual population. A technique termed profile maximum likelihood (PML) [45] is among the pioneers in the domain of property estimation. Further advancements have seen the integration of polynomial approximation in property estimation, as initiated by Acharya et al. [8]. This perspective is expanded by subsequent works, notably [53] and [54], which focused on entropy and NDV estimation. As noted by [23], NDV estimation can be interpreted as a weighted polynomial approximation, with other property estimation tasks following suit when considering sample frequencies.

While traditional research largely explores statistical techniques, the growth of machine learning has introduced new methodologies into property estimation, as evidenced by [52]. These learning-based approaches offer enhanced adaptability to distributional shifts, presenting an advantage over traditional counterparts. However, despite the practical efficacy of such methods, it is difficult to design the learning models for different property estimations one by one. In addition, we should ensure that the model can still have theoretical guarantees on the extreme data.

Motivation. Motivated by the limitation of existing studies, this paper aims to develop a learned polynomial-based method to unify the property estimation with theoretical guarantees and effectiveness. Our solution mainly utilizes the polynomial approximation framework, recasting property estimation as a weighted polynomial approximation problem. Our unified learning-based framework is based on the following key insights:

(1) Different property estimations correspond to different weighted polynomial approximations. We obtain a unified framework with the weighted polynomial approximation for estimating properties that employ a finite data distribution. We can easily modify the loss function and quickly solve a variety of weighted polynomial approximations with machine learning.
(2) We replace the polynomial method with a learning-based approach to derive weights for various property approximations. This maintains the optimal sampling complexity since we only modify the solving methods, not the theoretical framework of the polynomial approximation.

In summary, our methodology is guided by two core principles: leveraging polynomials for theoretical completeness and optimal sampling complexity and incorporating learning techniques to enhance practical performance through data distribution insights.

Overview of our method. Figure 1 shows the overview of our learning-based polynomial approximation framework for property estimation using the sampling frequency histogram. Our

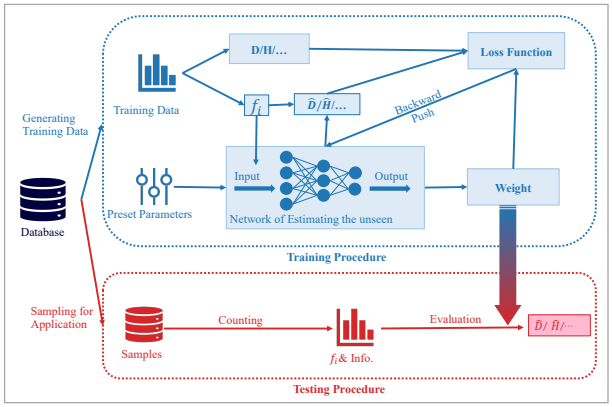


Fig. 1. Overview of Our Method.

primary goal is to determine the properties of other target columns using corresponding samples. Our learning process comprises two main components: *(i)* Training Phase: In this phase, we gather information, including sample frequency, sample rate, and workload of the samples, from known columns, which we use as features for training. These features are utilized to train a network responsible for estimating the unseen part and constructing the polynomial approximation. We design specific loss functions for different estimators and adjust polynomial weights based on property estimations for unseen elements and the precise computation of seen elements. Upon completing the training, we obtain stable training networks; *(ii)* Inference Phase: In this phase, we collect samples and record their frequencies. With the trained networks, we then estimate the property values of the target columns. This framework allows us to estimate properties efficiently despite having limited access to the database's columns for training.

Our contributions. This paper makes the following contributions.

- We integrate the problem of estimating properties of discrete distributions using samples by polynomials.
- We introduce the learnable polynomial method, a novel approach that can effectively handle diverse data distributions for property estimation of discrete distributions from samples.
- We provide a comprehensive theoretical analysis of the learnable polynomial approximation technique, particularly in the context of property estimation.
- To validate the efficacy of our approach, we conduct extensive experiments, demonstrating its validity and effectiveness

Paper organization. In Section 2, we formally define the problem of property estimation and related preliminaries. In Section 3, we present our property estimation algorithm and its theoretical analysis. We begin by establishing the connection between weighted polynomial approximation and property estimation. Then, we present the algorithm, which leverages weighted polynomial approximation in conjunction with machine learning for property estimation. In Section 4, we show our experimental results with respect to the performance and interoperability of algorithms. In Section 5, we review the related work of property estimation based on sampling. In Section 6, we offer a summary and conclusion of the paper.

Table 1. Table of notations.

Notation	Description
N	Size of the population in the raw data
D	NDV of the raw data
H	Shannon Entropy of the raw data
q	Sample rate
S	The size of the support set (abbreviated as the support size)
k	The upper bound of the support size
n	Size of the population in the sample
n_j	Element j appear n_j times in the sample
N_j	Element j appear N_j times in the population
d	NDV of samples
$F_1, \ldots, F_i$	Number of items appear i times in population
$f_0, f_1, \ldots, f_i$	Number of items appear i times in sample
$P_L(x)$	The polynomial with degree L
$\mathcal{X}_0$	The unseen part of $\mathcal{X}$

2 PRELIMINARIES

This section reviews some related and basic concepts. We summarize the frequently used notations in Table 1.

2.1 Problem Definition

When errors are allowed, sampling-based methods are extensively used to avoid incurring excessive I/O costs. With the population size (e.g., the number of records in the table) denoted as N and a predetermined sample rate q, existing property estimation solutions begin by sampling a subset of the data. Subsequently, they compute the frequency and the *frequency of frequency*, as defined below, for the samples of the columns.

Frequency of frequency. The *frequency of frequency* of the raw data consists of F_i, where F_i is the number of elements that occur i times in the raw data. F_i to compute the elements Given the sampled subset, the frequency of frequency is denoted as f_i, where f_i is the number of elements that occur i times in the sampled data. In particular, f_0 represents the number of elements that are not present in the samples. We also define the *frequency ratio* for the elements that occur i times as $\frac{i}{N}$.

Example 1 (*frequency of frequency*). *Given input data* $X = \{1, 2, 1, 3, 2, 4\}$*, the frequency of* X *is* $\{1 : 2, 2 : 2, 3 : 1, 4 : 1\}$ *and the* frequency of frequency *of* X *is* $\{2 : 2, 1 : 2\}$*. As a result, we have* $F_1 = 2, F_2 = 2$*. Using the* frequency of frequency *of* X*, the distinct values of* X *can be further calculated as* $D(X) = 2 + 2 = 4$*. If we sample from* X *obtaining the subset* $\{1, 1, 3\}$*, then the frequency of frequency for the sampled subset is* $f_0 = 2, f_1 = 1, f_2 = 1$*.*

Sampling-based property estimation. The definition of property estimation in [36] can be transferred to the field of the database. Given a column of the table, we focus on the property of population, which can be computed utilizing the population's frequency of frequency, expressed as follows:

$$\Psi = \sum_{i=1} F_i \cdot \psi\left(\frac{i}{N}\right).$$

Here, Ψ represents the desired property, $\psi(\cdot)$ is the function applied to the frequency ratio of each element in the property calculation, and F_i groups elements with the same frequency together. Our primary focus is on these additive elements, which are solely influenced by frequency. In this paper, we study three properties: NDV, entropy, and power sum, utilizing them as illustrative examples of our learning-based framework.

NDV Estimation. NDV is the number of the distinct values of the given raw data, denoted by D. The number of the distinct values of the sample is denoted by d. We give the definition of NDV in a mathematical way:

$$D = \mathbf{1}_{F_i} \cdot F_i,$$

where $\mathbf{1}_x$ is an indicative function, which is 1 when $x \neq 0$ and otherwise 0. In the NDV estimate, Ψ denotes D and $\psi(\frac{i}{N}) \coloneqq \mathbf{1}_{\frac{i}{N}}$.

Entropy Estimation. Entropy is a measure of uncertainty or information content in a random variable or a probability distribution. The entropy of the population is defined as follows:

$$H = -\sum_{i=1} \frac{i}{N} \log \frac{i}{N} F_i.$$

In entropy estimation, Ψ denotes H and $\psi\left(\frac{i}{N}\right) \coloneqq -\frac{i}{N} \log\left(\frac{i}{N}\right)$.

Power Sum. Given the *frequency of frequency* of the raw data consisting of F_i, the α-order power sum of the raw data is:

$$PS = \sum_{i=1} \left(\frac{i}{N}\right)^{\alpha} F_i.$$

In power sum, Ψ denotes PS and $\psi_{ps}\left(\frac{i}{N}\right) \coloneqq \left(\frac{i}{N}\right)^{\alpha}$.

Support Size Estimation. Support size estimation is an extension of the NDV estimation problem and inspires the design of other estimators. Support size estimation, as known in statistics, involves counting only unique elements whose sampled probability exceeds a certain threshold. These elements form the support set and have a sampled probability greater than $\frac{1}{k}$, where k represents a predetermined integer, also indicating the maximum number of elements the support set can contain. Let S denote the actual size of the support set and $\hat{S}$ be the estimate of S. The relative error of statistical methods in estimating the support set is defined as:

$$\mathcal{E}_{relative} = \mid S - \hat{S} \mid / S.$$

In the context of databases, when we set k equal to D, the problem of support size estimation becomes equivalent to NDV estimation.

Next, we revisit existing solutions for property estimation. Most existing solutions concentrate on specific problems such as NDV or entropy estimation. Therefore, we discuss the solutions for these two problems separately in Sections 2.2 and 2.3. Subsequently, we elaborate on existing learning-based solutions in Section 2.4.

2.2 Existing Solution for Sampling-based NDV

Currently, two main approaches are employed to estimate NDV using estimators. The first involves designing an estimator based on distribution hypotheses, while the second focuses on solving an optimization problem to derive the estimator. In this section, we provide an in-depth exploration of representative estimators. The process of designing an estimator begins with defining the error settings. Typically, denote the estimator of D as $\hat{D}$, the ratio error for estimators in database

applications is:

$$\mathcal{E}_{ratio} = \max\{D/\hat{D}, \hat{D}/D\}. \tag{1}$$

Charikar et al. [19] gives a negative result for the *ratio error*. It claims that for any estimators, counterexamples can be constructed such that the ratio error is at least

$$\mathcal{E}(D, \hat{D}) \geq \sqrt{\frac{N-n}{2n} \ln \frac{1}{\gamma}}.$$

Here, n is the population size of the sampled subset, and γ is the probability of failure with values exceeding $\exp^{-q}$, where q denotes the sample rate. The authors emphasize the significance of estimating unseen elements, i.e., f_0, through a challenging scenario posed by previous estimators. More precisely, this challenge involves distinguishing between the following two situations:

- N same elements.
- $(N-t)$ same elements and t distinct elements.

In the second scenario, the distinct number of samples is a good estimator only when the t distinct elements are all sampled. Otherwise, the estimator is likely to fail unless f_0 is well estimated. To handle the hard case, it is necessary to take the estimate f_0 into account. The authors analyze the optimal ratio error and derive the following estimator:

$$\hat{D}_{GEE} = d + \left(\sqrt{N/n} - 1\right) f_1, \tag{2}$$

where d is the NDV of the sample. Based on $\hat{D}_{GEE}$, Charikar et al. [19] propose a new estimator Adaptive Estimator (AE) to handle the low-skew data with guaranteed error. Different from $\hat{D}_{GEE}$, AE solving an equation about f_i and dynamically obtains a solution:

$$\hat{D}_{AE} = d + m - f_1 - f_2, \tag{3}$$

where m is the solution to the following equation:

$$m - f_1 - f_2 = f_1 \frac{\sum_{i=3} e^{-i} f_i + m e^{-(f_1+2f_2)/m}}{\sum_{i=3} i e^{-i} f_i + (f_1+2f_2) e^{-(f_1+2f_2)/m}}.$$

Another line of research work derives NDV estimators by solving optimization problems, with a particular focus on estimators based on polynomial approximation. For instance, Wu et al. [54] introduce the estimator for NDV as follows. For a given L, its estimator is

$$\hat{D}_{WY} = \sum_{j=1}^{L} g_L(j) f_j + \sum_{j>L} f_j, \tag{4}$$

where g_L is defined as

$$g_L(j) = \begin{cases} a_j j! + 1, & if\, j \leq L, \\ 1, & otherwise. \end{cases} \tag{5}$$

Here, the authors set $L \triangleq \lfloor c_0 \log k \rfloor$ with k being the input integer of support size estimation, $c_0 \approx 0.558$ and $a_0 = -1$ to ensure $g_L(0) = 0$. Chien et al. [22] highlight that the problem can be redefined as a regularization-based exponentially weighted Chebyshev approximation problem. While these existing estimators yield a collection of theoretically interpretable estimators through the sampling procedure, their practical performance often falls short of expectations.

Table 2. Estimators and their property.

Property	Estimator	Expression	Linear Estimator	Unseen	Using q	Distribution
NDV	$\hat{D}_{Plug-in}$	d	✓	✗	✗	✗
	$\hat{D}_{GEE}$ [19]	$d + \left(\sqrt{N/n} - 1\right) f_1$	✓	✓	✓	✗
	$\hat{D}_{Chao2}$ [18]	$d + \frac{f_1(f_1-1)}{2(f_2+1)}$	✗	✓	✗	✗
	$\hat{D}_{GT}$ [32]	$d + \sum_{i=1}(-1)^{i+1} t^i f_i$	✓	✓	✗	✗
	$\hat{D}_{AE}$ [19]	$d + m - f_1 - f_2$	✗	✓	✓	✗
	$\hat{D}_{WY}$ [54]	$\sum_{j=1}^{L} g_L(j) f_j + \sum_{j>L} f_j$	✓	✓	✗	✗
	$\hat{D}_{WD}$ [52]	$\exp(\sigma(W(\cdots \log(f_i \bigoplus features))$	✗	✗	✓	✓
	$\hat{D}_{our}$	$d + \sum_{j=1}^{L} b_j f_j$	✓	✓	✓	✓
Entropy	$\hat{H}_{Plug-in}$	$\sum_{i=1} f_i \frac{i}{n} \log \frac{n}{i}$	✓	✗	✗	✗
	$\hat{H}_{MM}$ [44]	$\hat{H}_{plug-in} + \frac{n-1}{2N}$	✗	✓	✓	✗
	$\hat{H}_{WY}$ [53]	$\sum_{j>L} f_j (\frac{j}{n} \log \frac{n}{j} + \frac{1}{2n}) + \sum_{j=1}^{L} g'_L(j) f_j$	✓	✓	✗	✗
	Ours	$\sum_{j>L} f_j \frac{j}{n} \log \frac{n}{j} + \sum_{j=1}^{L} b_j f_j$	✓	✓	✓	✓
Power sum	Ours	$\sum_{j>L} f_j \left(\frac{j}{n}\right)^{\alpha} + \sum_{j=1}^{L} b_j f_j$	✓	✓	✓	✓
Distance to uniformity	Ours	$\sum_{j>L} f_j \left(\frac{j}{n} - \frac{1}{k}\right) + \sum_{j=1}^{L} b_j f_j$	✓	✓	✓	✓
Normalized support coverage	Ours	$\sum_{j>L} f_j \frac{1-(1-j/n)^N}{N} + \sum_{j=1}^{L} b_j f_j$	✓	✓	✓	✓

Remarks: "Linear Estimator" refers to whether the estimator is a linear estimator. "Unseen" refers to whether the estimator is designed by considering the unsampled elements. "Using q" refers to whether the estimator contains the sampling probability q. The "distribution" refers to whether the estimator can adapt to different distributions.

2.3 Existing Solution for Sampling-based Entropy Estimation

Similar to NDV estimation, entropy estimation can be approached using both traditional and optimization methods. In the case of entropy estimation, previous research typically employs square error as a metric to assess the performance of estimators:

$$\mathcal{E}_{MSE} = (H - \hat{H})^2, \tag{6}$$

where $\hat{H}$ denotes the estimator of Shannon entropy. Aiming at the square error, Miller-Madow et al. [44] design the estimator:

$$\hat{H}_{MM} = \sum_{i=1} f_i \frac{i}{n} \log \frac{n}{i} + \frac{n-1}{2N}. \tag{7}$$

In Equation 7, it replaces the true probability with empirical frequency ratio for the estimation of entropy, while the term $\frac{n-1}{2N}$ is used to correct for bias. In addition to traditional statistics, polynomial estimation can also be applied to entropy estimation. Wu et al. [53] propose the entropy estimator based on polynomial approximation. The authors also divide the entropy into two parts by a fixed parameter L' and use the following estimator:

$$\hat{H}_{WY} = \sum_{j>L'} f_j \left(\frac{j}{n} \log \frac{n}{j} + \frac{1}{2n}\right) + \sum_{j=1}^{L'} g'_{L'}(j) f_j, \tag{8}$$

where L' is set to $\log k$ with k being the input integer of support size estimation, and g' represents the approximate polynomial used in entropy estimation. The rationale behind this equation is that the high-frequency part is estimated in the form of $\hat{H}_{MM}$, while the low-frequency part is estimated using g'.

2.4 Learning-based Estimation

The estimators listed above are designed with heuristics. Recently, learning-based strategies have been developed to adapt flexibly to data distribution. Generally, f_i is the main input of the property estimation, therefore Wu et al. [52] use f_i to learn the NDV estimator. We denote the learning-based estimator in [52] by $\hat{D}_{WD}$. As an overview, we divide the learning process into feature selection, model design, and the purpose of the loss function.

Training data and features. Learning-based methods require the inputs of fixed-dimensional features. However, the *frequency of frequency* usually differs between samples. As a solution, learning-based methods employ a cut-off in *frequency of frequency* as a remedy. For example, Wu et al. [52] use f_i with $i \leq 100$ as the input. To maintain efficiency, learning-based methods do not sample the raw data every time during the training process. Instead, they save a portion of F_i from the raw data for training and use random numbers from the binomial distribution as sampling. By changing the random number, a large amount of training data can be generated quickly.

Model design. The objective of learning-based models is to establish a connection between data features and NDV. According to existing studies, [32], F_i can be expressed as a linear combination of f_i. Wu et al. [52] employ a sequence of linear layers along with LeakyReLU as the activation function to enhance the model's representational capacity. In contrast to polynomial approximations, learning-based methods accommodate non-linear functions. While they preserve some linear relationships, this approach can diminish the physical prior knowledge of the model. Conversely, our goal is to design a more elegant and explainable model.

Loss function. The loss function in learning-based models controls the learning error. Wu et al. [52] use the loss function to regularize the model. Chien et al. [22] design the loss function aiming to minimize the bias and variance of estimation.

2.5 Summaries of Estimators

We list common property estimators in Table 2. A series of property estimators belong to the family of linear estimators. The linear addition of property estimations based on the frequency of frequency aligns with the interrelationship among elements. The key to property estimation lies in accurately estimating elements that have not been sampled. The main idea of existing property estimators is to *(i)* use the frequency ratio of the sampled portion to estimate the real probability for property estimation; *(ii)* approximate the bias between the unseen portion and the actual property. Different methods resolve these two from different perspectives. Building upon this idea, we can combine polynomial and neural network approaches to approximate this bias and address all properties collectively. It is noteworthy that most existing estimators do not take data distribution into account. Although Charikar et al. [19] adapt low-skew and high-skew data to design AE with guarantee error, AE still can not handle complex data distribution. Learning-based methods leverage data distribution to achieve superior results but lack explainability. Among all the estimators, our algorithm can be used flexibly to estimate different properties and enjoy desirable theoretical guarantees while injecting the learning techniques.

3 ALGORITHM

In this section, we provide an in-depth exploration of our framework. We begin by introducing the relationship between property estimation and polynomial approximation. Subsequently, we present how to use weighted polynomials for NDV estimation. Following that, we present a learning-based framework designed to estimate various additive properties, exemplified by entropy and power sum. Lastly, we provide a brief analysis of our algorithm.

3.1 Polynomials and Property estimation

NDV estimation. We begin by studying the relationship between polynomial and NDV estimation. We denote N_i as the number of occurrences of element i in the data. We model the sampling process as n independent random events using the Binomial distribution. The expectation of the *frequency of frequency* of sample, f_t is then:

$$\mathbf{E}[f_t] = \sum_{i=1}^{D} \binom{n}{t} \left(\frac{N_i}{N}\right)^t \left(1 - \frac{N_i}{N}\right)^{n-t}, \tag{9}$$

where $\frac{N_i}{N}$ is the probability to sample element i. Following the definition of linear estimators, the coefficient of f_t is set to a_t, i.e.:

$$\hat{D} = \sum_{t=1}^{L} a_t f_t + \sum_{t>L} f_t.$$

Here, we follow previous work and use L as a parameter to control the order of the *frequency of frequency* that we aim to learn for estimation. Note that L can also be interpreted as the gap between high-frequency and low-frequency elements. By inserting Equation (9) into linear estimator $\hat{D}$, we obtain the expectation of the basic estimator:

$$\mathbf{E}\left[\hat{D}\right] = E\left[\sum_{t=1} a_t f_t\right] = \sum_{t\geq 1}^{L} a_t \sum_{i=1}^{D} \binom{n}{t} \left(\frac{N_i}{N}\right)^t \left(1 - \frac{N_i}{N}\right)^{n-t} + \sum_{t>L} f_t.$$

Here, when $t > L$, we set $a_t = 1$. To explain, when the frequency of an element is above a certain threshold, we assume that it will be sampled when the sample size is sufficiently large and the more challenging part is to estimate for elements with low frequency. The bias of the property estimation comes mainly from the element that is not sampled, that is, f_0. However, to estimate the contribution of unsampled records in proper values, it is sufficient to use the sampled records. Next, we focus on the part of unseen elements, which, in the case of the NDV, is the estimation of the number of unseen elements, f_0. To do this, we take f_t as the input feature and aim to learn a linear estimator $\hat{f_0}$ of f_0 where the coefficient to be learned for f_t is denoted as b_t. Then, the estimator of f_0 is

$$\hat{f_0} = \sum_{t=1}^{L} b_t f_t. \tag{10}$$

By putting Equation (9) into Equation 10, we have that:

$$\mathbf{E}[f_0] = \sum_{t=1}^{L} b_t f_t = \sum_{t=1}^{L} b_t \sum_{i=1}^{D} \binom{n}{t} \left(\frac{N_i}{N}\right)^t \left(1 - \frac{N_i}{N}\right)^{n-t}.$$

Let $\mathcal{E}_D$ and $\mathcal{E}_{f_0}$ denote the error of estimating NDV, D and the number of unseen elements, f_0, respectively. According to the definition of D and f_i, we have $D = \sum_{i=1} \mathbf{1}(N_i) = \sum_{j=1} F_j$. Therefore, the error of D, $\mathcal{E}_D$ can be determined as follows:

$$\mathcal{E}_D = \sum_{i=1}^{D} \left(\sum_{t=1}^{L} \binom{n}{t} \left(\frac{N_i}{N}\right)^t \left(1 - \frac{N_i}{N}\right)^{n-t} a_t - 1 \right).$$

Merging elements with the same frequency, we have:

$$\mathcal{E}_D = \sum_{j=1} \left(\sum_{t=1}^{L} \binom{n}{t} \left(\frac{j}{N}\right)^t \left(1 - \frac{j}{N}\right)^{n-t} a_t - 1 \right) F_j. \tag{11}$$

For NDV estimation, the bias essentially arises from the unsampled elements, namely $\mathcal{E}_D = \mathcal{E}_{f_0}$. Note that $\hat{D} = \sum_{t=1}^{L} a_t f_t + \sum_{t>L} f_t$, $\hat{f_0} = \sum_{t=1}^{L} b_t f_t$, and $\hat{D} - \hat{f_0} = \sum_{t\geq 1} f_t$, we have that $a_t = b_t + 1$. Also, for $t > L$ since $a_t = 1$, we have that $b_t = 0$ for $t > L$. Thus, the above equation can also be expressed with the coefficient of b_t. The relationship between a_t and b_t indicates that the tasks of estimating unseen elements and NDV estimation are equivalent. For f_0, using the Binomial distribution, we can also derive its expected value:

$$\mathbf{E}[f_0] = \sum_{j=1} \left(1 - \frac{j}{N}\right)^n F_j. \tag{12}$$

Using Equation (12) and following the same technique in the analysis of $\mathcal{E}_D$, we have the following:

$$\mathcal{E}_{f_0} = \sum_{j=1} \left(\sum_{t=1}^{L} \binom{n}{t} \left(\frac{j}{N}\right)^t \left(1 - \frac{j}{N}\right)^{n-t} b_t - \left(1 - \frac{j}{N}\right)^n \right) F_j.$$

Obtaining an L-order polynomial approximation formula by simplifying the equation above by merging the elements with the same frequency, we have the following formula:

$$\mathcal{E}_{f_0} = \sum_{j=1} \left[\left(\sum_{t=1}^{L} Poly(N, n, j, t) b_t - 1 \right) F_j \left(1 - \frac{j}{N}\right)^n \right], \tag{13}$$

where $Poly(N, n, j, t) = \binom{n}{t} \left(\frac{j}{N-j}\right)^t$. When N, n is fixed, the value of $Poly(N, n, j, t)$ can be directly calculated. Since both $\mathcal{E}_D$ and $\mathcal{E}_{f_0}$ can be expressed with polynomials using b_t, the problem is hence how to determine the value of b_t.

Since we can only get experimental values (integers) for f_i but not the exact expected value (decimals), the relationship between b_t and f_t is not exactly a simple linear relationship. To increase the interpretability of b_t, we treat b_t as the function of f_t. In concrete terms, we compute b_t by a machine learning model with f_t as input. With this adjustment, we can preserve the linear correlation of f_t and NDV. At the same time, we can introduce more interpretive possibilities in b_t for the estimator. According to Equation (13), the problem of estimating the NDV is also amenable to approximation by polynomials. Following the same process of deduction, we can obtain the other estimators for different properties.

Entropy estimation. We next present how to extend to entropy estimation. We use the frequency ratio of samples to replace the frequency ratio of the population to estimate the entropy of the sampled elements. Also, because we have parameterized b_t with f_t, a learning-based neural network can then learn the coefficient b_t of f_t that is required for the calculation of entropy. The bias of the entropy estimator will be:

$$\mathcal{E}_{entropy} = \sum_{j=1} \left[\left(\sum_{t=1}^{L} Poly(N, n, j, t) b_t - \frac{j}{N} \log \frac{N}{j} \right) F_j \left(1 - \frac{j}{N}\right)^n \right]. \tag{14}$$

We also need to solve the polynomial approximation problem for entropy estimation. We first propose to use machine learning to unify the problem of property estimation. With the parameterization of b_t, we can use gradient descent to solve the polynomial approximation more conveniently and stably.

Drawing from the analyses above, the general form of bias in estimating the properties of other unseen elements can be summarized in the following:

$$\mathcal{E}_{\Phi} = \sum_{j=1} \left[\left(\sum_{t=1}^{L} Poly(N, n, j, t) b_t - \psi\left(\frac{j}{N}\right) \right) F_j \left(1 - \frac{j}{N}\right)^n \right]. \tag{15}$$

Following the same processes, we can deduce the bias of α-order power sum as:

$$\mathcal{E}_{PS} = \sum_{j=1} \left[\left(\sum_{t=1}^{L} Poly(N, n, j, t) b_t - \left(\frac{j}{N}\right)^{\alpha} \right) F_j \left(1 - \frac{j}{N}\right)^n \right]. \tag{16}$$

From Equation (15), we see that for different data distributions, F_j varies with the distribution. When multiplied by $(1 - \frac{j}{N})^n$, the result is also an uncertain value. For each j, this represents a polynomial approximation problem. In essence, it is a weighted polynomial approximation problem where the weight is given by $F_j(1 - \frac{j}{N})^n$. $F_j(1 - j/N)^n$ represents the expected number of times of an element that appears j times in population but is not sampled. In [54], they turn $F_j(1 - j/N)^n$ into $P_L \cdot \exp(-np_i)$ with the Poissonization technique, where p_i is the expectation probability of the elements being sampled and P_L is the polynomials.

3.2 Learning-based NDV Estimation with Weighted Polynomial Approximation

In this part, we describe the details of the learning-based polynomial approximation algorithm. We introduce the algorithm to learn the NDV as an example. The NDV estimation of weighted polynomials is actually the case of the most basic property estimation. The other property estimation can follow the same algorithm with Equation (13) with tiny modifications.

Based on Equation (13), we consider the coefficient b_t as the function with respect to f_i. Then, the coefficients b_t for $t \leq L$ is a set of values that can be learned according to different distributions and parameters rather than a fixed set of values. However, for the previous polynomial approximation methods [22, 54], the authors solve a fixed coefficient b_t. Hence, these previous solutions do not generalize to different distributions. For the previous learning-based NDV estimation [52], the authors rarely explore the relationship between property and f_i, and thus lack theoretical guarantees.

Algorithm 1 shows the pseudo-code of our proposed learning-based NVD estimation algorithm. We explain our algorithm in three parts: preparation, training, and loss.

- **Preparation**. According to Equation (13), we pre-compute and save the value of polynomials, $Poly(N, n, j, t)$ for training and estimation. In the previous learning-based NDV estimation [52], the authors take the logarithm of the input f_i directly to make features, $\log D$ and $\log \hat{D}$ at the same scale and introduce the non-linearity. The different scales of features and NDV are due to the mismatch between the data scales of the binomial coefficients and frequencies. We provide a clearer physical interpretation by pre-computing the value of polynomials and adjusting the scale of the data in advance. We initialize weights as an all-one vector. When the training data has varied inputs, the weights will be as close to the definition, namely $F_j \left(1 - \frac{j}{N}\right)^n$. This is reflected in the numerical values that have a numerical value for the high frequencies and 0 for the low frequencies. We also verify this in our experiments.
- **Training**. We use a simple two-layer linear activation network. Compared to the complex multi-layer neural network structure in [52], our network is simpler and requires fewer parameters. The introduction of the polynomial minimizes unnecessary assumptions and improves the training efficiency.

Algorithm 1: NDV Estimation with Polynomials

Input: f_j : The *frequency of frequency* of samples.
N: The size of the population in the raw data.
n: The size of the sample.
L: The order of polynomial approximation.
λ: Hyperparameters for the weighted sum of the loss function.
W: The weight of polynomials.
Output: The estimation of NDV.

Initialize the weight of the polynomials as **1**.
Initialize the two-layer neural network $\mathcal{N}et$ of the NDV estimation.
Pre-compute $Poly(N, n, j, t)$.
while *Training* **do**
  $b_t \leftarrow \mathcal{N}et(f_j)$.
  bias_loss $\leftarrow \sum_{j=1} \sum_{t=1} (b_t \cdot Poly(N, n, j, t) - 1) \cdot w_j$.
  $\hat{f_0} \leftarrow Relu\,(\sum_{t=1} b_t \cdot f_t)$.
  loss = lossfun($\hat{f_0}, D, d$)+λ·bias_loss.
  Backward Push.
  Update $\mathcal{N}et$ and weight W.
while *Evaluating* **do**
  $b_t \leftarrow \mathcal{N}et(f_j)$.
  $\hat{f_0} \leftarrow Relu\,(\sum_{t=1} b_t \cdot f_t)$.
  Return $\hat{f_0} + \sum_{t=1} f_t$.

- **Loss**. We divide the loss into two parts and balance them with adjustable penalty parameters, which are the actual estimation error and the bias loss as in Equation (13). For the actual estimation error of NDV, we adopt the loss given in [52]. An advantage of our framework is that it is easier to combine different losses. We do not need to define specific loss functions for different property estimation tasks.

3.3 Learning-based Property Estimation with Weighted Polynomial Approximation

In this part, we describe how the framework for NDV estimation with polynomials can be adapted to estimate different properties. We take the two most representative properties: entropy and power sum, as examples following [8, 36].

We provide the algorithm for estimating property using polynomials in Algorithm 2. In the following part, we discuss the similarities and dissimilarities when estimating different properties with detailed explanations.

- **Similarities.** The general framework of training follows the previous pattern. The property estimates for those elements in the seen part use the sampled frequency as probabilities to estimate their property values, which is equivalent to maximum likelihood estimation. The unseen parts are estimated with the machine learning model, which still uses a two-layer activation network. We use the activation networks and polynomials to construct a compound loss, which consists of estimated loss and bias loss, to get an estimator of the property of the unseen part. Typically, the function ψ used for property estimation is non-negative. This is consistent with activation networks where the final result only needs to be fed into the ReLU function.

Algorithm 2: Property Estimation with Polynomials

Input: f_j : The *frequency of frequency* of samples.
N: The size of the population in the raw data.
n: The size of the sample.
L: The order of polynomial approximation.
λ: Hyperparameters for the weighted sum of the loss function.
W: The weight of Polynomials .
Output: The estimation of entropy.

Initialize the weight of the polynomials as **1**.
Initialize the two-layer linear activation layer Network $\mathcal{N}et$ of the property estimation.
Pre-compute $Poly(N, n, j, t)$.
while *Training* **do**
  $b_t \leftarrow \mathcal{N}et(f_j)$.
  bias_loss $\leftarrow \sum_{j=1} \sum_{t=1} \left(b_t \cdot Poly(N, n, j, t) - \psi\left(\frac{t}{N}\right)\right) \cdot w_j$.
  $\hat{\Psi}_0 \leftarrow Relu\left(\sum_{t=1} b_t \cdot f_t\right)$.
  loss = lossfun($\hat{\Psi}_0, f_t, \Psi$) + λ·bias_loss.
  Backward Push.
  Update $\mathcal{N}et$ and weight W.
while *Evaluating* **do**
  $b_t \leftarrow \mathcal{N}et(f_j)$.
  $\hat{\Psi}_0 \leftarrow Relu\left(\sum_{t=1} b_t \cdot f_t\right)$.
  Return $\hat{\Psi}_0 + \sum_{t=1} f_t \psi(\frac{t}{n})$.

- **Dissimilarities**. Compared to counting values like NDV, when the features and property Ψ's scale is mismatched, we can also take logarithm like [52]. For example, entropy is a function of probability, which is inconsistent with the scale of input f_i. Therefore, we can take the logarithm of the input for entropy estimation. We adapt the function of bias loss based on polynomial approximation theory, namely Equation (15). As we introduce in previous sections, different standards of error will be used for different properties. For the different properties and evaluation errors, we can use different estimated loss functions. In particular, we directly use the square error for the estimation loss of entropy and the absolute error for the power sum. Since we are directly adding estimated errors and the polynomial approximation bias, we are detecting whether the scale is matched. If not, we need to consider which of them needs to be processed to match the scale.

After summarizing our above framework, we next present a theoretical analysis of our learning-based solutions.

3.4 Theoretical Analysis

In this section, we give some theoretical analysis of our algorithm. We follow the setting of most property estimation research works. For the unknown distribution with *frequency of frequency* F_i and property Ψ, the mean squared error (MSE) for the estimator $\hat{\Psi}(f_i)$, where f_i is the *frequency of frequency* of samples with n samples, is

$$\mathcal{E}_n\left(\hat{\Psi}, \Psi, f_i\right) \coloneqq \mathbf{E}_{f_i}\left(\Psi - \hat{\Psi}\left(f_i\right)\right)^2,$$

Table 3. Error bounds on smallest sample complexity for studied properties.

Property	Lower bound	Upper bound
NDV	$\frac{D}{\log D}\log^2\frac{1}{\varepsilon}$	$\frac{D}{\log D}\log^2\frac{1}{\varepsilon}$
Entropy	$\frac{D}{\varepsilon\log D}+\frac{\log^2 D}{\varepsilon^2}\log\frac{1}{\delta}$	$\frac{D}{\varepsilon\log D}+\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)^{1+\beta}$
α Power sum	$\frac{D^{\frac{1}{\alpha}}}{\varepsilon^{\frac{1}{\alpha}}\log D}+\frac{D^{2-2\alpha}}{\varepsilon^2}\log\frac{1}{\delta}$	$\frac{D^{\frac{1}{\alpha}}}{\varepsilon^{\frac{1}{\alpha}}\log D}+\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)^{\frac{1+\beta}{2\alpha-1}}$

Remarks: β can be an fixed absolute constant in $(0,1)$. The lower and upper bounds for power sum hold for $\alpha\in(1/2,1)$.

where $\mathcal{E}_n$ denotes the errors over n samples.

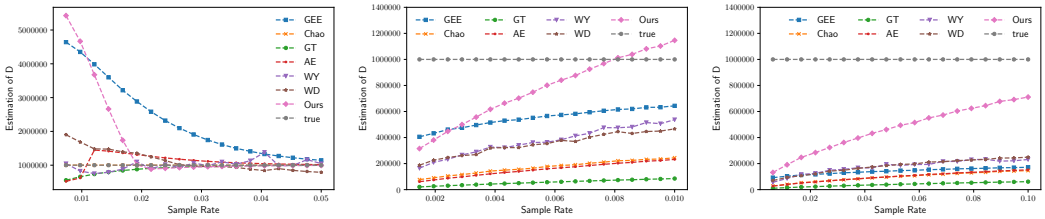
We aim to obtain the estimator $\hat{\Psi}$ of Ψ with the accuracy ε, confidence $1-\delta$ and smallest sample complexity $C(\hat{\Psi},\varepsilon,\delta)$. Hao et al. [36] have given an analysis of the different properties. We are essentially using machine learning methods to compute the polynomial approximations, still adhering to the conclusions of [36]. However, the conclusion in [36] assumes an infinite distribution while we focus on the setting of a table with a limited size. Next, we show how to extend their conclusion to our setting.

High-level description. At a high level, the polynomial approximation methods replace the underlying function ψ with a polynomial and use the empirical frequency as input [36]. The bias mainly comes from the polynomial approximation for the elements unsampled. The ability to control the bias becomes crucial for the accuracy of the estimation. As [36] states, when we need to consider whether the main errors come from the empirical plug-in estimation or from the unbiased polynomial approximation, we have to analyze case-by-case which seriously affects the efficiency. Machine learning makes it easier to generate polynomial approximations of different properties. Besides, we do not have to balance the errors case-by-case by controlling the ratio of the loss function.

Following the settings of the property function, the condition for L-Lipschitz property estimation in [36] (Theorem 1 therein) is also satisfied in the case of polynomial approximations based on learning. In the context of databases, D is the value of the upper bound on the support size, k. We replace the k in [36] Table 2 with D and obtain the sample complexity bounds for the properties in the databases in Table 3.

Analysis for the parameters. We analyze the values of the parameters, especially the range of polynomial approximations. First, we determine the complexity of the dimension of weights. When the expectation of sampled times about an element with frequency j is sufficiently small, we do not use it to estimate f_0. Concretely, when the expectation threshold is δ_1, the upper bound of summation of j is $O(\frac{1}{q}\log\frac{Nq}{\delta_1})$. In the experiment, we can reduce the dimension of approximate weight according to the specific distribution when the weights with large indexes are all 0. To maintain the $L-$Lipschitz property, we keep the polynomial order settings the same as the previous polynomial approximation [36, 54], namely $L=O(\log D)$. The main bottleneck in our learning process is the process of computing the bias loss, related to the dimension of the matrix formed by the pre-computed polynomials, i.e., the product of the upper bound of the two summations of the bias, $O(\frac{\log D}{q}\log\frac{Nq}{\delta_1})$.

Analysis for Algorithm 2. Compared with the previous polynomial approximation research works [36, 53, 54], we follow the previous parameter settings and assumptions, the difference is



(a) Uniform (b) Zipf 1.2 (c) Zipf 1.5

Fig. 2. The result of NDV estimation vs sample rate q on synthetic datasets. (a) Uniform distribution. (b,c) Zipf distribution with skewness factors γ 1.2 and 1.5 respectively.

only that the method of approximating the optimal polynomial coefficients is different. Although the result learned from the neural network may not be an exact optimal polynomial approximation, it is still a bounded function over $[0, 1]$, so it still satisfies Theorem 4 and 5 in [36].

4 EXPERIMENT

In this section, we experimentally evaluate our methods. They are compared to other baselines in three aspects:

- Performance. How precise are our learning-based estimators in both synthetic and real-world datasets?
- Interpretability. What parameters do we really learn?
- Ablation study. How does the introduction of polynomials influence the performance?

4.1 Experiment Setup

Hardware. All our experiments are performed on a Linux machine with 40 Intel(R) Xeon(R) Silver 4114 CPU at 2.20GHz, an NVIDIA RTX A6000 GPU (48GB memory), and 512 GB of RAM.

Datasets. We evaluate our experiments on both synthetic and real-world datasets. To ensure fairness, we use the same training data in [52]. The data generation process is provided in [3] To be fair, we re-train the model of $\hat{D}_{WD}$ according to the source code with the newly generated data. For the synthetic data, we consider data independently sampled from certain distributions under the same conditions as in [54], which are:

- *Uniform distribution.* $p_i = \frac{1}{k}$, where k is equal to 10^6 with the population size, $N = 10^8$.
- *Zipfian distribution.* In the database, the size of the column is finite, so we fixed $N = 10^8$ items from Zipfian distribution belonging to $D = 10^6$ elements, whose $p_i \propto i^{-\gamma}$. We set the skewness factor γ as 1.2 and 1.5.

We also include four datasets from [52] for comparison, which are:

- *Airline* [1]: Summary statistics of airline departures from 1987 to 2013 with 10.0M rows and 10 columns.
- *Kasandr* [49]: Collection for recommendation systems that record the behavior of customers of the European leader in e-commerce advertising with 15.8M rows and 7 columns.
- *NCVR* [2]: North Carolina voter registration data with 8.3M rows and 71 columns.
- *SSB* [47]: The star schema benchmark. We use the fact table with a scaling factor of 50, resulting in 300M rows and 17 columns.

Methods evaluated. We compare two stable methods derived from traditional estimators. The remaining estimators are intended for certain distributions. Besides, referring to the experiments of [52], GEE and Chao's estimators outperform other classic methods. We therefore focus our comparisons on these statistics. We include NDV estimation with Chebyshev polynomial approximation as a baseline, namely $\hat{D}_{WY}$. We compare with the public version of the learning-based NDV estimation [4, 52]. We list the summary of the methods in the experiment in the following:

- NDV
  - *GEE* [19]. [19] provide a theoretical lower bound of ratio error within a constant factor. This estimator uses the geometric mean to handle the hard case of NDV estimation.
  - *Chao* [46]. Chao's estimator is derived by approximating the coverage as $1 - f_1/n$ and assuming the population size is infinity. To avoid it blowing up when $f_2 = 0$, we use the expression in Equation (18).
  - *Shlosser* [48]. Shlosser's estimator is derived from the assumption: $\mathbf{E}[f_i]/\mathbf{E}[f_1] \approx F_i/F_1$ when the population is large. The method is constructed for language dictionaries and performs well when most elements appear once on average.
  - *GT* [33]. We use the expression in Equation (20). We set the parameter θ of Equation (20) as 0.1.
  - *AE* [19]. AE estimator is an upgraded version of the GEE estimator. It has guaranteed error and can handle low- and high-skew data.We follow the setting in [52] by using the classic Brent's method [15] to find the non-linear equation of AE estimator.
  - *WY* [54]. Following the name in [27], we abbreviate the estimator using the Chebyshev polynomial approximation as **W**Y. The parameters of the estimator follow the settings in [54].
  - *WD* [52]. The learning estimator for NDV is designed by [52]. We abbreviate its name as Learning. We use the public version[4] using the same training datasets provided by itself.
  - *Ours.* Our learning-based NDV estimation method with polynomials in Algorithm 1.
- Entropy
  - *Plug-in.* Plug-in estimator [29] directly uses the sampling frequency as the true probability for estimation. Plug-in estimator performs well in practice when most elements are sampled and is also a good baseline with sufficient interpretability.
  - *MM* [44]. Miller-Madow's estimator reduces the bias of the plug-in estimator. The Miller-Madow estimator is also a frequently used method in entropy estimation.
  - *WY* [53]. [53] gives the way of using polynomials approximation to estimate entropy. We directly use the code provided by [53] [1].
  - *Ours.* Our learning-base entropy estimation method with polynomials approximation in Algorithm 2.
- α-order Power Sum. We fix the order α of the power sum to 0.5.
  - *Plug-in.* We use the frequency of the samples replacing the true probability to estimate the α-order power sum. This estimator is a lower bound of the power sum and a basic baseline.
  - *Ours.* Our learning-base power sum estimation method with polynomials approximation in Algorithm 2.

Setup for estimators. According to [19, 20], low sample rates result in the failure of all estimators. So we set the sample rates ranging from 0.001 to 0.01. To deal with randomness coming from sampling, we repeat experiments 10 times and report the median of results. For the parameters of baselines, we carefully follow the setting of the original papers. For WY's estimator [54]. We set $c_0 = 0.45, c_1 = 0.5$, which is optimized to yield the best convergence rate in Proposition 3 in [54]. We apply a degree-7 polynomial in all experiments for our methods. The initial values of weight

[1]https://github.com/Albuso0/entropy.git

are set to 1. We use ReLU as the activation function and the learning rate to be 0.001. All of our methods are implemented with PyTorch.

Performance metric. As mentioned above, due to different background needs, different estimators utilize different errors for evaluation. We use different performance metrics for different estimators. For NDV estimation and power sum, we use the ratio error as the performance metric, which is defined in Equation (1). We use the absolute error for entropy estimation:

$$\mathbf{E}_{abs} = |H - \hat{H}|.$$

We do not use the square error since this error is small on all compared methods. We hence use the absolute error, which tends to be larger and can signify the difference.

4.2 Performance

In this section, we report the performance of our learning-based estimators with polynomials varying on different properties estimation and datasets. In all tables, boldface indicates the best results.

Table 4. The performance of different NDV estimators (Ratio Error).

Methods	Kasandr				Airline				SSB				NCVR				Average
	0.001	0.005	0.01	Time(s)	0.001	0.005	0.01	Time(s)	0.001	0.005	0.01	Time(s)	0.001	0.005	0.01	Time(s)	
GEE	2.455	1.480	**1.335**	1.0	2.754	1.388	1.205	0.3	2.770	1.825	1.578	2.3	5.589	2.385	1.906	4.4	2.223
Chao	3.828	2.219	1.855	0.9	1.452	1.238	1.195	0.3	**1.069**	**1.053**	**1.046**	2.2	11.450	3.983	7.640	4.2	3.169
WY	4.143	1.642	1.370	8.4	**1.269**	1.345	1.323	3.0	4.019	1.538	1.268	20.5	8.641	2.774	2.401	37.6	2.645
GT	30.515	7.768	4.672	2.4	1.604	1.328	1.262	0.7	35.945	7.866	4.360	5.8	67.466	15.980	9.106	9.7	15.656
Shlosser	7.618	4.348	3.321	48.0	5.524	1.155	**1.074**	12.7	25.570	8.335	5.461	118.4	14.555	1.608	**1.274**	187.5	6.654
AE	33.231	7.494	4.427	109.8	1.293	1.156	1.133	12.2	39.452	8.575	4.710	295.8	59.450	12.617	6.979	221.8	15.043
WD	2.342	1.883	1.730	0.2	1.608	1.249	1.279	0.2	1.574	1.478	1.293	0.4	4.125	1.984	1.745	1.8	1.857
Ours	**2.085**	**1.297**	1.395	3.0	1.343	**1.102**	1.084	2.9	2.447	1.646	1.781	6.7	**2.796**	**1.478**	1.310	25.3	**1.647**

Table 5. The performance of different entropy estimators (Absolute Error).

Methods	Kasandr				Airline				SSB				NCVR				Average
	0.001	0.005	0.01	Time(s)	0.001	0.005	0.01	Time(s)	0.001	0.005	0.01	Time(s)	0.001	0.005	0.01	Time(s)	
Plug-in	1.151	0.651	0.475	0.046	0.025	0.007	0.004	0.077	1.502	0.901	0.679	0.033	0.529	0.358	0.301	0.315	0.549
MM	0.972	0.505	0.346	0.045	**0.008**	**0.003**	**0.002**	0.077	1.293	0.723	0.518	0.031	0.463	0.314	0.261	0.307	0.451
WY	19.040	3.774	1.887	0.108	20.467	4.087	2.044	0.169	17.266	3.367	1.678	0.178	21.782	4.220	2.068	0.836	8.473
Ours	**0.499**	**0.250**	**0.204**	2.589	0.025	0.007	0.004	1.971	**0.191**	**0.045**	**0.037**	6.355	**0.268**	**0.177**	**0.173**	17.115	**0.157**

NDV estimation performance. Figure 2 (a,b,c) displays the curve of the estimated values of NDV as the sample rate increases from the different estimators on synthetic datasets. The performance of Shlosser's estimator in this experiment is not shown in Figure 2 due to its extremely poor performance. According to Figure 2 (a), both traditional and our estimator perform well in uniform distribution. They both converge to the ground-truth NDV as the sampling rate increases. However, when dealing with the Skew distribution and Zipf distribution, the performance of traditional methods is not satisfactory. With the skewness factor increase, traditional estimators become difficult to converge. Since they are related to the assumption of distribution at the beginning of their design, their estimation accuracy becomes worse as the distribution becomes diverse. For example, Shlosser's estimator is effective when each data item occurs very few times (satisfying $\frac{\mathbf{E}[f_i]}{\mathbf{E}[f_1]} = \frac{F_i}{F_1}$), but it is particularly ineffective in the uniform distribution. The estimate of learning-based methods are not strictly linear about the sampling rate q during evaluation, but oscillate above and below around the true value with sample rate changing. In Figure 2 (b), our curve initially rises, falls, and finally approaches the true value. This is due to the generalization error arising

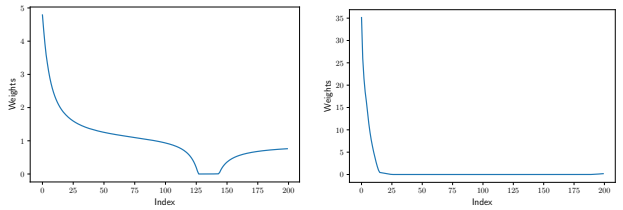
Table 6. The performance of power sum estimation ($\alpha = 0.5$ Ratio Error).

Datasets	q	Plug-in	Time(s)	Ours	Times(s)
Kasandr	0.001	3.182	0.273	**2.222**	1.965
	0.005	1.886		**1.371**	
	0.01	1.584		**1.206**	
Airlines	0.001	1.041	0.415	**1.027**	1.720
	0.005	**1.015**		1.016	
	0.01	**1.009**		1.017	
SSB	0.001	3.843	0.082	**2.281**	4.818
	0.005	2.092		**1.203**	
	0.01	1.693		**1.057**	
NCVR	0.001	3.100	1.277	**2.219**	14.161
	0.005	1.965		**1.456**	
	0.01	1.679		**1.296**	

from the inconsistency between the sampling rates chosen during the generation of training data and those of the test data. When generating training data, the proportion is relatively high in the 0.001-0.01 sampling rate range, leading to a slight overfitting. The weights may not be suitable for the same data at higher sampling rates. A new model could be retrained with higher q, for data at higher sampling rates. In database applications, a fixed sampling rate is often preset. We can focus on this sampling rate range and generate training data accordingly.

In the real-world datasets, we set the evaluation standard to be ratio error in Equation (1). We conduct experiments at the sample rate 0.01, 0.005, and 0.001. We repeat the experiments 10 times to avoid outliers and take the median values. We display the performance of different NDV estimators on real-world datasets in Table 4. On average, our method achieves the best results in the four datasets. WY's estimator has a good theoretical guarantee in the support size estimation for infinite discrete distributions. However, in real database applications, WY's estimator performs the worst. Using the method of learning can effectively avoid the manual adjustment of the distribution and better apply the theory to reality. It is quite difficult to surpass some traditional methods on certain datasets, especially when we do not train for a specific distribution, because they could be the optimal solution for certain specific distributions at the beginning of the design. However, the learning-based method is relatively more stable. Besides, in practical use, traditional estimators may face unacceptably large errors. For example, Chao's estimator performs very well on SSB datasets, but in NCVR, with the sampling rate being 0.001, the ratio error of Chao's estimator is unacceptable. While the AE estimator aims to mitigate the effects of both high and low skewness inherent in data distributions through optimization methods, its performance is suboptimal with real-world data. Designing estimators tailored to specific data distributions might enhance estimation accuracy for those particular distributions. However, this approach often results in limited generalizability. In contrast, our method does not rely on pre-existing assumptions about the data distribution. Instead, it adapts to variations in the data distribution by learning directly from the data itself.

Compared to traditional methods, learning-based methods have smaller estimation errors and fewer extreme cases, making them the preferred choice in practical applications. Our method is superior to the results of the learning-based method by Wu et al. [52] in performance and has a clear advantage in training time (as we will see later) and interpretability. Through NDV estimation, we mainly show that by introducing polynomial approximation, we can achieve similar or better results than learning NDV with a more complex design.



(a) Weights of $F_j = \frac{N}{j}, N =$ 100000, $j \sim Uniform(50, 55)$. (b) Weights of our final model

Fig. 3. Weights in different training data.

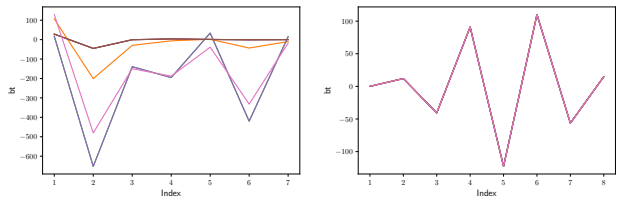
Performance of entropy and power sum estimation. We only evaluate the entropy estimation on the four real-world datasets. Compared to the previous learning-based methods [52], our design is free of complex network design. Table 5 shows the performance of different entropy estimators. Our algorithm is superior to other methods in most cases, except for the Airline dataset. The results also show that traditional estimators are still competitive on some real-world data. Learning algorithms perform well overall and are the most stable and robust in real-world datasets. WY's estimator uses a fixed coefficient, and it is difficult to deal with real-world data. By combining polynomial approximation and learning techniques, we can better apply the theory of polynomial property estimation to practical applications. Through entropy estimation, we can see that a learning-based polynomial estimator is more stable in practice than traditional estimators based on fixed distribution assumptions.

We further show that learning-based polynomial approximation methods can be easily applied to power sum estimation. Table 6 shows the performance of our algorithm on power sum estimation. We use the ratio error as the evaluation criterion. To our knowledge, none of the previous solutions for power sum estimation [36] can deal with an arbitrary choice of α, say $\alpha = 0.5$. Hence, we only compare our learning-based solution against the plug-in estimator with $\alpha = 0.5$. In power sum estimation, we add the logarithm of the absolute error loss and bias loss from polynomials as a loss function. Compared to the complex loss function design in [52], our simple loss function design can be quickly applied to different property estimations and obtain good results. We evaluate the power sum estimation for sample rates of 0.001 and 0.005. As shown in Table 6, even at a small sample rate, our method brings certain improvements and is very stable.

4.3 Interpretability and Ablation Study

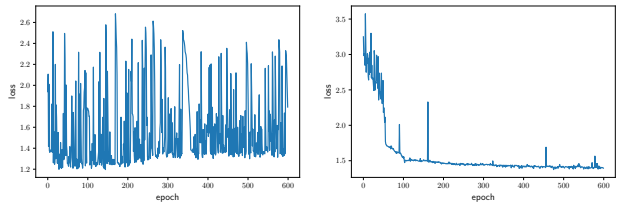
In this section, we experimentally explain what our model is actually learning. Compared to the original polynomial method, we can give the reason why our method can achieve better performance than the original polynomial approximation methods.

Training on different distribution. Different from simply stacking multiple linear activation layers in [52], the parameters learned from our algorithm can reflect part of the information of the training data. As Figure 3 (a) shows, when our training data is fixed on some single peak data (for example, $F_j = \frac{N}{j}, N = 100000$,
$j \sim Uniform(50, 55)$), the weights will have a set of abnormal valley values to reduce the bias loss of data. Figure 3 (b) shows the weight distribution of the NDV estimation model obtained by our algorithm. It can be roughly seen that the weight of the part with greater influence on



(a) b_t of our estimator in Kasandr. (b) b_t of WY's estimator

Fig. 4. Dynamically adjusted parameters.



(a) Training loss without polynomial approximation. (b) Training loss with polynomial approximation.

Fig. 5. The bias loss of polynomial approximation helps the model converge.

f_0 is concentrated on the part with lower frequency, which also satisfies Equation (13). Setting the weight of the high-frequency part to 0 is the best choice, which also explains why the weight dimension L does not need to be very large.

Dynamically adjusted linear estimator. According to the linear estimator mentioned in [54], the polynomial approximation theory is indeed very significant in the specific data distribution. In essence, property estimation can be transformed into a weighted polynomial approximation problem. However, what function is the property estimation essentially approximating, apart from the property value estimated? Can it also capture the initial distribution? Figure 4 (b) shows the linear coefficients b_t of WY's estimator. For any input, the coefficients of WY's estimator are fixed, which is not adaptable to distributional changes. Figure 4 (a) shows the coefficients of Kasandr dynamically learned by our algorithm. Each line corresponds to a different column of final learned approximations. WY's estimator only uses a fixed set of approximate coefficients for different tasks unless each data is manually adjusted. In most cases, Chebyshev polynomial approximation is not the best approximation to real-world data. We automatically and dynamically adjust, according to different observed data, learn the data distribution information, and get different linear parameters to estimate b_t, which has a high ability to express information and higher application value.

Ablation study. We also explore the performance of the learning model like [52], which simply stacks two linear activation layers only. By removing the bias part of the loss function, we can see that the gradient update does not optimize the polynomial approximation parts. In this way, we are able to control the influence of polynomial approximation. When we eliminate the bias loss part of the loss function, the weights change to a set of horizontal straight lines with a value of 1. Undoubtedly, the effect of such a model makes it difficult to satisfy the demand. We draw the change in its loss function value in Figure 5 (a). It can be seen that such a model is difficult

to converge. Figure 5 (b) illustrates the variation of the loss function for our model utilizing the polynomial approximation. Our method reaches convergence after a certain epoch.

Training time. In terms of model structure design, $\hat{D}_{WD}$ [52] use seven linear activation layers, while we only use two. In terms of input features, $\hat{D}_{WD}$ [52] use f_1 to f_{100} as input features, while we only use f_1 to f_{70} by default. Due to the simplicity of our model structure, convergence is relatively faster. Our estimator obtains a lower error with a shorter training time under a relatively simple network, which is precisely caused by the fact that the model implies more physical knowledge. Additionally, our model is streamlined and does not require a GPU for training. Using a CPU for training is adequate and does not notably extend the training time. In the final performance experiment, the training time of the $\hat{D}_{WD}$ is 6037s. In comparison, the preparation time of our method (Pre-compute in the third line of the Algorithm 1) is 392s and the training time is only 341 seconds on a GPU or 371 seconds on a CPU. Compared to previously learned methods, the training time of our method improves by an order of magnitude.

5 RELATED WORKS

Property estimation is typically addressed separately for each specific sub-problem, with tailored solutions designed to tackle these unique challenges. Common properties of interest in the context of discrete data distributions include NDV (Number of Distinct Values), entropy, coverage, power sum, and more. Among them, the exploration of property estimation first commences with the focus on NDV estimation [30, 32]. Notably, Valiant et al. [51] identify a critical connection between NDV and entropy. This pivotal insight not only propels their work on solving systems of linear equations specifically for NDV [50] but also catalyzes its extension to entropy estimation [51]. Acharya et al.[8] and Hao et al.[36] provide further consolidation by summarizing methods for property estimation [53, 54] that rely on polynomial approximations. In the following sections, we delve into related works, examining them from the specific perspectives of NDV and entropy estimation.

5.1 NDV Estimation

The problem of estimating the number of distinct values (NDV) has been widely discussed. Sampling-based NDV estimation can be classified into two primary categories: statistics and database. From the statistical perspective, the data is assumed to come from an infinite discrete distribution. The problem then becomes the support size estimation. For support size estimation, we seek to use as few samples as possible. Valiant and Valiant [51] have shown that the sample complexity could be reduced to $\frac{k}{\varepsilon^2 \log k}$, where ε is the relative error. Later, polynomial-based methods for support size estimation appear. Based on the Chebyshev polynomial approximation, Wu et al. [54] expand the result to $\frac{k}{\log k} \log^2 \frac{1}{2\varepsilon}$, to obtain a min-max lower bound. Based on [54], several works try to improve the precision [22, 27] of support size estimation. Chien et al. [22] use both the variance and bias and the traditional optimal methods to solve the problem. According to [54], the problem of estimating the support size can be solved by Chebyshev polynomial approximation with the Poissonization technique.

On the contrary to the statistical methods, the amount of data is finite when utilized in the databases. Part of the methods [12, 31] maintain a small data structure and scan the whole table to obtain the NDV of data. However, when the amount of data is large, scanning all the elements requires a high computation cost and I/O expenses [41]. Therefore, using sampling to solve the NDV estimation becomes an appealing solution.

Early studies on classical estimators [17, 32, 48] focus on the estimation of f_0. These estimators can be formally defined as:

$$\hat{D} = d + \mathcal{F}(f_1, f_2, ..., f_i), \tag{17}$$

where d is the NDV of sample and $\mathcal{F}(f_1, f_2, ..., f_i)$ is a function of *frequency of frequency* f_i.

A typical representative for estimating f_0 is Chao's Estimator [17], $\hat{D}_{Chao} = d + \frac{f_1^2}{2f_2}$. Ozsoyoglu et al. [46] use it to estimate the NDV in the database. The original version of Chao's estimator blows up when $f_2 = 0$. A bias-corrected version of Chao's Estimator is proposed in [18]:

$$\hat{D}_{Chao2} = d + \frac{f_1(f_1 - 1)}{2(f_2 + 1)}. \tag{18}$$

GEE and Chao's estimators estimate NDV using an approximation of f_0. High-frequency elements are counted by d. Low-frequency partial elements account for the unseen elements in the sample. The remaining low-frequency elements are counted using low *frequency of frequency* such as f_1, f_2.

The family of linear estimators. Following the research of Good [32], a series of works focuses on the linear estimator family. These estimators utilize a linear combination of f_i's functions, which can be written as follows:

$$\hat{D} = \sum_i g(i) f_i. \tag{19}$$

We provide three examples of linear estimators in the following:

- Plug-in Estimator: $\hat{D} = \sum f_i = d$.
- Good-Toulmin Estimator [33]: For some $\theta > 0$,

$$\hat{D}_{GT} = d + \sum_i (-1)^{i+1} \theta^i f_i. \tag{20}$$

- Efron-Thisted Estimator [29]: For some $\theta > 0$ and $J \in \mathbf{N}$,

$$\hat{D}_{ET} = d + \sum_{j=1}^{J} (-1)^{j+1} \theta^j b_j f_j,$$

where $b_j = \mathbf{Pr}[Binomial(J, 1/(\theta + 1)) \geq j]$.

The utilization of the linear structure can effectively ensure the interpretability of estimators. Therefore, linear estimators are favored in both NDV estimation and entropy estimation. Note that polynomial approximation methods are also linear estimators.

Besides the exploration of traditional methods, many database methods seek to enhance the performance of NDV estimators in practice via learning [52]. Since the columns of a database are data-dependent, learning the distribution of input helps the algorithm to adapt to different data distributions. However, learning-based methods generally lack theoretical guarantee. It is desirable if traditional methods could be combined with learning-based methods to work in practice with theoretical guarantees.

Cardinality estimation. Recently, A series of works use machine learning to estimate cardinality [9, 25, 26, 42, 43, 55]. These studies vary in their methodologies: some utilize query words as features to estimate query cardinality [25], while others employ the joint distribution of queries for the same purpose [37, 55]. The outcomes of these cardinality estimations are predominantly used to enhance query optimization processes [55].

Our approach also incorporates learning methods for estimating NDV. Furthermore, NDV estimation plays a crucial role in facilitating more intricate forms of cardinality estimation. For instance, Spark and PostgreSQL (PG) leverages NDV to appraise the cardinalities of queries [5, 6].

5.2 Entropy Estimation

From the perspective of computer science, estimating entropy is the basis for information theory [13, 34]. In natural language processing, the entropy of context is used to measure the amount of information in a text.

For the fixed distribution, the maximum likelihood entropy estimators are given by Antos et al. [10] and Strong et al. [40]. To reduce the bias due to convergence rate, Miller et al. [44] provide a bias correction estimator. The use of re-sampling to reduce variance is relatively common in statistics, where it is represented by jackknife and boosting. Efron et al. [28] provide the jackknife version of the entropy estimator. Next, we give a mathematical expression of these classical estimators.

- Plug-in Estimator [10]: Using the frequency ratio of the samples to replace the frequency ratio of the population,

$$\hat{H}_{plug-in} = -\sum_{i=1} p_i \log p_i = \sum_{i=1} f_i \frac{i}{n} \log \frac{n}{i}. \tag{21}$$

- The jackknifed Estimator [28]: Reducing the variance of estimators using resampling techniques,

$$\hat{H}_{JK} = N\hat{H}_{plug-in} - \frac{N-1}{N} \sum_{j=1}^{N} \hat{H}_{plug-in\setminus j}, \tag{22}$$

  where $\hat{H}_{plug-in\setminus j}$ represent the plug-in estimators removing the jth element.

Valiant et al. [50] estimate the entropy by solving the linear programming, which proves that the minimal sample size for consistent entropy estimation is $\Theta(\frac{k}{\log k})$, where k is the upper bound of the support size. Wu et al. [53] use polynomial approximation to achieve optimal minimax mean square error rate $(\frac{k}{n \log k})^2 + \frac{\log^2 k}{n}$, where n is the sample size. Hao et al. [36] unify the sample-optimal property estimation in Near-Linear time and give the lower and upper bound of different property estimations.

While the previous methods enjoy desirable theoretical guarantees [21], their performance in real-world applications is questionable. Methods [36, 53] that use the polynomial to estimate properties are not generalizable to different data distributions. The nonlinear estimators [36] use Remez algorithm to fit in varied distributions, but it is still unstable and difficult to optimize. Introducing learning into entropy estimation seems to be a promising solution to address the above difficulties, yet to the best of our knowledge, there is no existing work addressing entropy estimation through the application of machine learning techniques.

6 CONCLUSION

In this paper, we focus on how to obtain a property estimator that is learnable with theoretical design. We propose a framework utilizing learning techniques and polynomial approximation to estimate different properties. Unlike previous learning-based methods, our method contains the physical explanation of parameters and has theoretical guarantees about the error. Experiments on various synthetic and real-world datasets demonstrate the effectiveness of our estimator. In future work, we try to enhance the expressiveness of our estimators and extend our algorithm to incorporate other estimators. Besides, an interesting direction is how to incrementally learn new models as the distribution shifts.

ACKNOWLEDGMENTS

This research was supported in part by National Natural Science Foundation of China (No. U2241212, No. 61932001), by Beijing Natural Science Foundation (No. 4222028), by Beijing Outstanding Young Scientist Program No.BJJWZYJH012019100020098, and by Alibaba Group through Alibaba

Innovative Research Program. Sibo Wang was supported by Hong Kong RGC GRF (Grant No. 14217322) and Hong Kong ITC ITF (Grant No.MRP/071/20X). We also wish to acknowledge the support provided by the fund for building world-class universities (disciplines) of Renmin University of China, by the funds from Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education, from Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China, from Public Policy and Decision-making Research Lab of Renmin University of China, and from Public Computing Cloud, Renmin University of China.

REFERENCES

[1] 2020. Airlines Departure Delay. https://www.openml.org/d/42728.
[2] 2020. Voter Registration Statistics. https://www.ncsbe.gov/results-data/voterregistration-data.
[3] 2021. Synthetic data generator. https://github.com/wurenzhi/learned_ndv_estimator.git.
[4] 2022. Github. https://github.com/wurenzhi/learned_ndv_estimator.git.
[5] 2023. Source Code of Spark. https://github.com/apache/spark/blob/master/sql/catalyst/src/main/scala/org/apache/spark/sql/catalyst/plans/logical/statsEstimation/JoinEstimation.scala.
[6] 2024. Source Code of PostgreSQL. https://github.com/postgres/postgres/blob/master/src/backend/optimizer/plan/analyzejoins.c.
[7] Jayadev Acharya, Hirakendu Das, Alon Orlitsky, and Ananda Theertha Suresh. 2017. A unified maximum likelihood approach for estimating symmetric properties of discrete distributions. In *International Conference on Machine Learning*. PMLR, 11–21.
[8] Jayadev Acharya, Hirakendu Das, Alon Orlitsky, and Ananda Theertha Suresh. 2017. A unified maximum likelihood approach for estimating symmetric properties of discrete distributions. In *International Conference on Machine Learning*. PMLR, 11–21.
[9] Christos Anagnostopoulos and Peter Triantafillou. 2015. Learning to accurately count with query-driven predictive analytics. In *2015 IEEE international conference on big data (big data)*. IEEE, 14–23.
[10] András Antos and Ioannis Kontoyiannis. 2001. Convergence properties of functional estimates for discrete distributions. *Random Structures & Algorithms* 19, 3-4 (2001), 163–193.
[11] Aryaman Arora, Clara Meister, and Ryan Cotterell. 2022. Estimating the Entropy of Linguistic Distributions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 175–195.
[12] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. 2002. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science*. Springer, 1–10.
[13] Tuğkan Batu, Sanjoy Dasgupta, Ravi Kumar, and Ronitt Rubinfeld. 2002. The complexity of approximating entropy. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 678–687.
[14] Srikanth Bellamkonda, Hua-Gang Li, Unmesh Jagtap, Yali Zhu, Vince Liang, and Thierry Cruanes. 2013. Adaptive and big data scale parallel execution in oracle. *Proceedings of the VLDB Endowment* 6, 11 (2013), 1102–1113.
[15] Richard P Brent. 2013. *Algorithms for minimization without derivatives*. Courier Corporation.
[16] John Bunge and Michael Fitzpatrick. 1993. Estimating the number of species: a review. *J. Amer. Statist. Assoc.* 88, 421 (1993), 364–373.
[17] Anne Chao. 1984. Nonparametric estimation of the number of classes in a population. *Scandinavian Journal of statistics* (1984), 265–270.
[18] Anne Chao and TJ Shen. 2010. User's guide for program SPADE (Species prediction and diversity estimation). *Taiwan: National Tsing Hua University* (2010).
[19] Moses Charikar, Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. 2000. Towards estimation error guarantees for distinct values. In *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 268–279.
[20] Surajit Chaudhuri, Gautam Das, and Utkarsh Srivastava. 2004. Effective use of block-level sampling in statistics estimation. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. 287–298.
[21] Xingguang Chen and Sibo Wang. 2021. Efficient approximate algorithms for empirical entropy and mutual information. In *Proceedings of the 2021 International Conference on Management of Data*. 274–286.
[22] Eli Chien, Olgica Milenkovic, and Angelia Nedich. 2021. Support Estimation with Sampling Artifacts and Errors. In *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 244–249.
[23] I Chien. 2019. Regularized weighted Chebyshev approximations for support estimation. (2019).
[24] Reuven Cohen and Yuval Nezri. 2019. Cardinality estimation in a virtualized network device using online machine learning. *IEEE/ACM Transactions on Networking* 27, 5 (2019), 2098–2110.
[25] Anshuman Dutt, Chi Wang, Vivek Narasayya, and Surajit Chaudhuri. 2020. Efficiently approximating selectivity functions using low overhead regression models. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2215–2228.
[26] Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek Narasayya, and Surajit Chaudhuri. 2019. Selectivity estimation for range predicates using lightweight models. *Proceedings of the VLDB Endowment* 12, 9 (2019), 1044–1057.
[27] Talya Eden, Piotr Indyk, Shyam Narayanan, Ronitt Rubinfeld, Sandeep Silwal, and Tal Wagner. 2021. Learning-based Support Estimation in Sublinear Time. In *International Conference on Learning Representations*.
[28] Bradley Efron and Charles Stein. 1981. The jackknife estimate of variance. *The Annals of Statistics* (1981), 586–596.
[29] Bradley Efron and Ronald Thisted. 1976. Estimating the number of unseen species: How many words did Shakespeare know? *Biometrika* 63, 3 (1976), 435–447.

[30] Ronald A Fisher, A Steven Corbet, and Carrington B Williams. 1943. The relation between the number of species and the number of individuals in a random sample of an animal population. *The Journal of Animal Ecology* (1943), 42–58.
[31] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*. Discrete Mathematics and Theoretical Computer Science, 137–156.
[32] Irving J Good. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika* 40, 3-4 (1953), 237–264.
[33] Irving J Good and George H Toulmin. 1956. The number of new species, and the increase in population coverage, when a sample is increased. *Biometrika* 43, 1-2 (1956), 45–63.
[34] Sudipto Guha, Andrew McGregor, and Suresh Venkatasubramanian. 2006. Streaming and sublinear approximation of entropy and information distances. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. 733–742.
[35] Yanjun Han, Jiantao Jiao, and Tsachy Weissman. 2018. Local moment matching: A unified methodology for symmetric functional estimation and distribution estimation under wasserstein distance. In *Conference On Learning Theory*. PMLR, 3189–3221.
[36] Yi Hao and Alon Orlitsky. 2019. Unified sample-optimal property estimation in near-linear time. *Advances in Neural Information Processing Systems* 32 (2019).
[37] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2019. Deepdb: Learn from data, not from queries! *arXiv preprint arXiv:1909.00607* (2019).
[38] Wen-Chi Hou, Gultekin Ozsoyoglu, and Baldeo K Taneja. 1988. Statistical estimators for relational algebra expressions. In *Proceedings of the seventh ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. 276–287.
[39] Jiantao Jiao, Yanjun Han, and Tsachy Weissman. 2018. Minimax estimation of the L_{1} distance. *IEEE Transactions on Information Theory* 64, 10 (2018), 6672–6706.
[40] Roland Koberle, Rob R De Ruyter Van Steveninck, and William Bialek. 1998. Entropy and information in neural spike trains. *Physical review letters* 80, 1 (1998), 197.
[41] Jiajun Li, Zhewei Wei, Bolin Ding, Xiening Dai, Lu Lu, and Jingren Zhou. 2022. Sampling-based Estimation of the Number of Distinct Values in Distributed Environment. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 893–903.
[42] Pengfei Li, Wenqing Wei, Rong Zhu, Bolin Ding, Jingren Zhou, and Hua Lu. 2023. ALECE: An Attention-based Learned Cardinality Estimator for SPJ Queries on Dynamic Workloads. *Proceedings of the VLDB Endowment* 17, 2 (2023), 197–210.
[43] Henry Liu, Mingbin Xu, Ziting Yu, Vincent Corvinelli, and Calisto Zuzarte. 2015. Cardinality estimation using neural networks. In *Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering*. 53–59.
[44] George Miller. 1955. Note on the bias of information estimates. *Information theory in psychology: Problems and methods* (1955).
[45] Alon Orlitsky, Narayana P Santhanam, Krishnamurthy Viswanathan, and Junan Zhang. 2004. On modeling profiles instead of values. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. 426–435.
[46] Gultekin Ozsoyoglu, Kaizheng Du, A Tjahjana, W-C Hou, and DY Rowland. 1991. On estimating COUNT, SUM, and AVERAGE relational algebra queries. In *Database and Expert Systems Applications*. Springer, 406–412.
[47] Patrick O'Neil, Elizabeth O'Neil, Xuedong Chen, and Stephen Revilak. 2009. The star schema benchmark and augmented fact table indexing. In *Technology Conference on Performance Evaluation and Benchmarking*. Springer, 237–252.
[48] A Shlosser. 1981. On estimation of the size of the dictionary of a long text on the basis of a sample. *Engineering Cybernetics* 19, 1 (1981), 97–102.
[49] Sumit Sidana, Charlotte Laclau, Massih R Amini, Gilles Vandelle, and André Bois-Crettez. 2017. KASANDR: a large-scale dataset with implicit feedback for recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1245–1248.
[50] Gregory Valiant and Paul Valiant. 2011. Estimating the unseen: an n/log (n)-sample estimator for entropy and support size, shown optimal via new CLTs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*. 685–694.
[51] Paul Valiant and Gregory Valiant. 2013. Estimating the Unseen: Improved Estimators for Entropy and other Properties.. In *NIPS*. 2157–2165.
[52] Renzhi Wu, Bolin Ding, Xu Chu, Zhewei Wei, Xiening Dai, Tao Guan, and Jingren Zhou. 2021. Learning to be a statistician: learned estimator for number of distinct values. *Proceedings of the VLDB Endowment* 15, 2 (2021), 272–284.
[53] Yihong Wu and Pengkun Yang. 2016. Minimax rates of entropy estimation on large alphabets via best polynomial approximation. *IEEE Transactions on Information Theory* 62, 6 (2016), 3702–3720.

[54] Yihong Wu and Pengkun Yang. 2019. Chebyshev polynomials, moment matching, and optimal estimation of the unseen. *The Annals of Statistics* 47, 2 (2019), 857–883.

[55] Rong Zhu, Ziniu Wu, Yuxing Han, Kai Zeng, Andreas Pfadler, Zhengping Qian, Jingren Zhou, and Bin Cui. 2021. FLAT: fast, lightweight and accurate method for cardinality estimation. *Proceedings of the VLDB Endowment* 14, 9 (2021), 1489–1502.

Received 15 October 2023; revised 19 January 2024; accepted 23 February 2024