Cardinality Estimation: Is Machine Learning a Silver Bullet?

Beibin Li[1], Yao Lu[2], Chi Wang[2], Srikanth Kandula[2]
University of Washington[1], Microsoft[2]

ABSTRACT

Cardinality estimation (CE) aims for high accuracy, small storage, fast building and low query answering latency. We analyze the upper error bounds of random uniform sampling for single-table CE and use them as the accuracy target for machine learning (ML)-based CE. Our analysis indicates that ML-based CE exhibits no Pareto advantage over random uniform sampling but provides a tradeoff among the metrics of interest. We outline such tradeoffs and point out the scenarios when ML-based CE can be useful and when sampling can help.

1. INTRODUCTION

Substantial prior work focused on cardinality estimation (CE) of a given query predicate, which in turn enables various database tasks such as costing query plans in optimizers ([2, 4, 3, 5, 10, 7, 18]). The state-of-the-art in practice is to use histograms [14], sketches such as Count-Min and HyperLogLog [5]. Recent work on Machine Learning (ML)-based CE learns dataset-specific models to answer CE queries [11, 21, 16, 19, 8]. An ideal CE solution should build its data structure and answer queries quickly with high accuracy.

Current ML-based CE solutions have shown gains in some aspects but have not achieved a Pareto improvement over simple alternatives such as random uniform sampling [20]. For example, DeepDB [11] and Naru [21] demonstrate promising accuracy but require large storage in MBs and building latency in tens to a few hundred minutes; in contrast, sampling has minor building overhead and may already provide accurate estimates for some predicates at smaller storage. ML is *not* a silver bullet to CE.

To better understand this, we analyze the error bounds of random uniform sampling in Section 2. We compare different metrics-of-interest (storage, accuracy, building and query latency) between ML-based methods and sampling in Section 3. Our analyses show that random uniform sampling already provides tight upper error bounds when the storage is adequate or when predicates have high cardinality.

3rd International Workshop on Applied AI for Database Systems and Applications (AIDB'21), August 20, 2021, Copenhagen, Denmark.

When ML-based CE outperforms classic solutions is an open question to the literature and database practitioners. We point out the usefulness criteria for ML-based CE to trade among the metrics of interest. Lastly, we describe in Section 3.1 a simple but effective solution to use samples as a gate before ML-based CE.

Nevertheless, the theoretical analyses in this paper show single-table results. We defer analyses for CE over multiple tables to [12]; we also refer the readers to [20] for empirical study and comparisons of recent ML-based CE solutions.

2. BOUNDS OF RANDOM SAMPLING

To compare the accuracy of different CE methods, we leverage the theoretical error bounds of random uniform sampling. Following recent CE work [11, 21, 16, 8], we analyze the Q-error to evaluate the estimation accuracy:

$$\texttt{Q-error} = \max(\frac{\texttt{gt}}{\texttt{pred}}, \frac{\texttt{pred}}{\texttt{gt}}),$$

where `pred` is the predicted cardinality and `gt` is the ground truth. There have been numerous bound analyses on random uniform sampling in the last a few decades, but few have covered the Q-error metric in the context of CE.

We introduce the problem setup in Section 2.1, analyze random uniform sampling with replacement in Section 2.2, and random uniform sampling without replacement in Section 2.3. The error bounds will be used in Section 3 to indicate when ML-based CE can be useful. Analyses with detailed proofs can be found in our technical report [17].

2.1 Problem setup

Let t be a table with n rows and m columns, and $\hat{t}$ be k rows sampled uniformly at random from t with or without replacement, where $k \leq n$. Suppose a total of $X = pn$ rows from the table t satisfy a given predicate, where $p \in [0, 1]$ represents the probability that a row in the table satisfies the predicate. Hence, X is the cardinality of the input query predicate. Cardinality estimation, i.e., estimating X given a predicate, can be formulated as an application of Binomial distribution (Sum of Independent Bernoulli Trials), in which each random variable (row) takes the value of 1/0 for satisfying the predicate or not.

Let $X = pn = \sum_{i=0}^{n} x_i$, where $x_i = 1$ if row i satisfies the predicate, and $\hat{X} = kp = \sum_{i=0}^{k} \hat{x}_i$. Hence, $\hat{X}$ is the cardinality of the sampled table $\hat{t}$ that satisfies the predicate. $\texttt{pred} = \frac{\hat{X}n}{k}$ estimates the cardinality X for table t. We use $\mu =$

$\mathsf{E}[\hat{X}] = kp$ to represent the expected number of rows in the sampled table $\hat{t}$ that satisfy the given predicate, with a population variance $\sigma^2 = p(1-p)$.

2.2 Sampling with replacement

Random uniform sampling with replacement follows the independent and identically distributed (i.i.d.) assumption and is widely applied in modern machine learning (e.g. bootstrap [9]). In this case, $\hat{t}$ is randomly sampled with replacement from t. The Chernoff Bound together with the Bernstein's Inequality give us a concise upper error bound with the Q-error metric.

Theorem 1. *For CE over single tables, the Q-error of random uniform sampling with replacement is bounded by*

$$\mathsf{P}(\textit{Q-error} \le q) \ge 1 - \quad - \quad, \textit{where}$$
$$= \min\left(\left(\frac{e^{q-1}}{q^q}\right)^{pk}, \exp\left(-\frac{k(pq-p)^2}{2\sigma^2 + 2(pq-p)/3}\right)\right),$$
$$= \min\left(\left(e^{(\frac{1}{q}-1)}q^{\frac{1}{q}}\right)^{pk}, \exp\left(-\frac{k(p-p/q)^2}{2\sigma^2 + 2(p-p/q)/3}\right)\right).$$

When P(Q-error $\le$ q) becomes negative (i.e. $\Omega + \Psi > 1$), we replace it with zero. Ω is the probability for over-estimation and Ψ is the probability for under-estimation.

From Theorem 1, we can see that the (target) error q is *agnostic* to the number of rows n. The bound is only relevant to the sample size k and ground truth selectivity p.

2.3 Sampling without replacement

When samples are drawn uniformly at random without replacement, we assume there are at least 2 sampled rows. For simplicity, we define

$$\rho = \begin{cases} 1-(k-1)/n & \text{if } k \le n/2 \\ (1-k/n)(1+1/k) & \text{if } k > n/2 \end{cases}$$

$$\zeta = \begin{cases} 4/3 + \sqrt{\frac{k(k-1)}{n(n-k+1)}} & \text{if } k \le n/2 \\ 4/3 + \sqrt{\frac{(n-k-1)(n-k)}{(k+1)n}} & \text{if } k > n/2. \end{cases}$$

Based on the Hoeffding-Serfling Inequality and the Bernstein-Serfling Inequality [1], we have:

Theorem 2. *For CE over single tables, the Q-error of random uniform sampling without replacement is bounded by*

$$\mathsf{P}(\textit{Q-error} \le q) \ge 1 - \quad - \quad, \textit{where}$$
$$= \min\left(2\exp\left(\frac{k}{\zeta^2}\left(\sqrt{2\zeta\rho\sigma^2(pq-p) + \rho^2\sigma^4} - (pq-p)\zeta - \sigma^2\rho\right)\right), \exp\left(-\frac{2k(pq-p)^2}{\rho}\right)\right)$$
$$= \min\left(2\exp\left(\frac{k}{\zeta^2}\left(\sqrt{2\zeta\rho\sigma^2(p-p/q) + \rho^2\sigma^4} - (p-p/q)\zeta - \sigma^2\rho\right)\right), \exp\left(-\frac{2k(p-p/q)^2}{\rho}\right)\right)$$

p	X	100 rows		1000 rows		10000 rows	
		R	NR	R	NR	R	NR
0.0002	166	0.00	0.00	0.00	0.00	0.00	0.00
0.0003	333	0.00	0.00	0.00	0.00	0.12	0.00
0.0008	833	0.00	0.00	0.00	0.00	0.68	0.00
0.0010	1000	0.00	0.00	0.00	0.00	0.76	0.00
0.0017	1666	0.00	0.00	0.00	0.00	0.92	0.42
0.0050	5000	0.00	0.00	0.39	0.00	1.00	0.96
0.0083	8333	0.00	0.00	0.68	0.00	1.00	1.00
0.0100	10000	0.00	0.00	0.76	0.00	1.00	1.00
0.1667	166666	0.92	0.75	1.00	1.00	1.00	1.00
0.3333	333333	0.99	1.00	1.00	1.00	1.00	1.00
0.5000	500000	1.00	1.00	1.00	1.00	1.00	1.00
1.0000	1000000	1.00	1.00	1.00	1.00	1.00	1.00

Table 1: Confidence that the Q-error is at most 2 (i.e., $q \le 2$) from Theorem 1 and Theorem 2. We randomly sample 100, 1000, or 10000 rows from 1 million rows. R: Sampling with Replacement; NR: Sampling without Replacement.

Visualization. Figure 1 shows a 3-D visualization of the probability that random uniform sampling is better than a given Q-error at a selectivity threshold. At a small sample size such as 100 or 1K, random uniform sampling with replacement already provides promising q-error and has a tight bound.

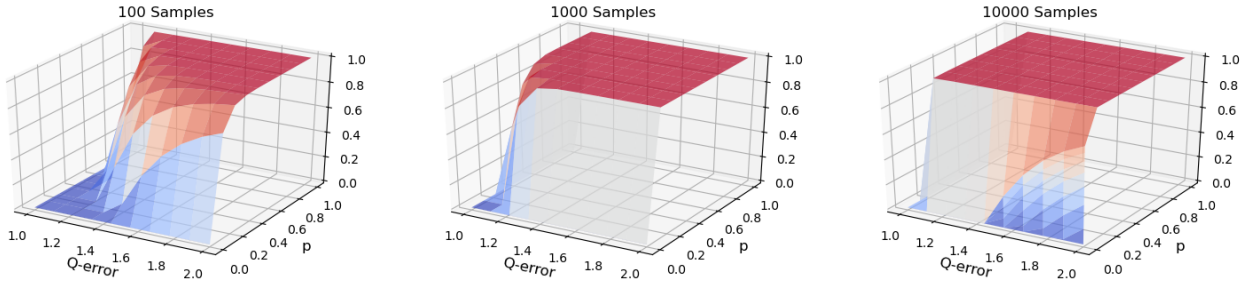
3. WHEN ML IS BETTER FOR CE?

Metrics of interest. To answer when ML is better for CE compared to simple alternatives such as sampling, metrics of interest include accuracy, query answering cost, and building cost. Storage used, on the other hand, is often less of a concern but has a direct impact on the accuracy and the costs; both are functions of the storage as will be shown in Table 2. For most solutions, more storage results in better accuracy but potentially larger building and query answering latency; for example, a sample that is as large as the original dataset provides the best accuracy but also the worst query latency. We discuss below scenarios when a storage budget s exists and also when s does not exist, trading accuracy with latency using storage as a handle.

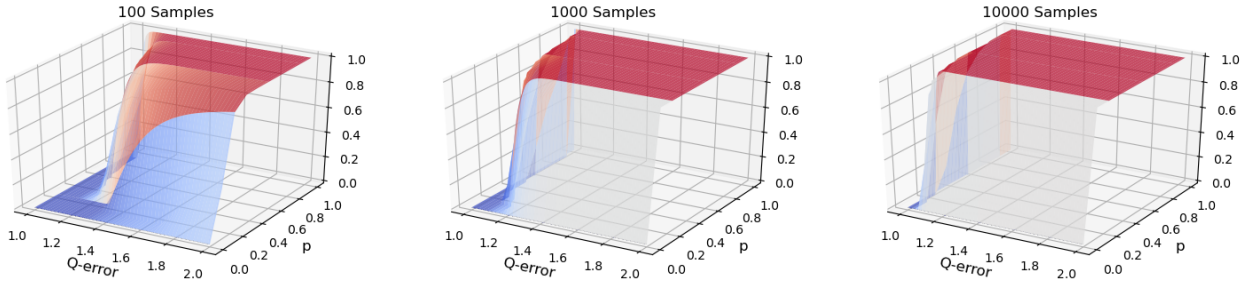
Modeling workload vs data. Current ML-based CE models learn from workload (e.g., LM [8] and MSCN [15]) or data (e.g., Naru [21] and DeepDB [11]). The former learns the mapping from predicates to true cardinality and hopes for unchanged workload between training and testing; predicates y from the training workload are required. The latter captures the distribution and multi-column correlations in data and is not trained with a predicate set (i.e., the model is *agnostic* to workload). These distinct learning strategies lead to different comparisons below.

Evaluating individual metrics at storage s. We first inspect how sampling, workload-driven ML models, and data-driven ML models behave on each metric mentioned above in Table 2 at storage budget s. We will compare the metrics at different storage budget choices afterwards.

Accuracy. We leverage the upper error bound for random sampling (Section 2) as the accuracy target for ML-based CE and plug in the storage budget s and the test predicates y or synthetic $\hat{y}$ (depending on if the test predicates are



(a) random uniform sampling with replacement.



(b) random uniform sampling without replacement.

Figure 1: 3D plotting of Q-errors for random uniform sampling with replacement (first row) and without replacement (second row) with 100 samples (left), 1K samples (middle), or 10K samples (right). We show the probability that random uniform sampling's error (z-axis) is better than the desired Q-error (y-axis) with a predicate ground truth selectivity p (x-axis).

known) to Theorem 1 or 2[1] which yields an upper error bound q on the predicates evaluated. ML-based solutions, however, often cannot provide theoretical bounds and we rely on empirical evaluation (i.e., `eval`() in Table 2) of the trained model M on y or $\hat{y}$ at storage s. An ML-based CE solution can be useful only when it provides better than q accuracy using the same storage s.

Notably, as discussed in Section 2, the upper error bounds of sampling converge quickly when the storage increases; in general, sampling has better accuracy than many current ML-based solutions at large storage or when predicates have high cardinality. Sampling and data-driven models are agnostic to the test workload y; when y is unknown, we defer picking synthetic $\hat{y}$ to later discussion.

Query answering cost. Learned CE solutions, regardless of workload- or data-driven, often use models with dense parameters (e.g., neural networks) and incur high query answering latency since multiple costly computations such as matrix multiplication need to be done on the entire storage s. In contrast, evaluating predicates on samples takes one pass over the storage and often can be accelerated using heuristics (e.g., short-circuiting for conjunctive predicates and ranking columns in terms of selectivity); the complexity is at most linear to s (when no acceleration is applied). Learned CE solutions with dense models are often slower than evaluating predicates on samples of the same storage. Indeed, efforts such as using GPUs and SIMD can accelerate inference of dense models; it is yet unclear from the literature which effort could lead to a clear win. To this end, if query answering latency is a major concern, the ML model has to be small or uses sparse data structures (e.g., trees

[1]The user or downstream application decides the confidence threshold on $\mathsf{P}(Q\text{-}error \leq q)$, e.g., 0.95.

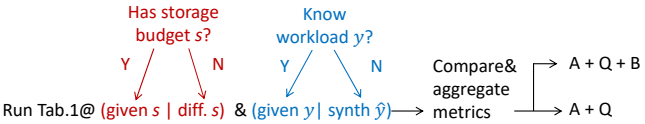


Figure 2: Picking a better CE solution. Given the storage budget s, we run Table 2 and choose a better solution by aggregating different metrics of interest including (A)ccuracy, (Q)uery answering cost and (B)uilding cost. When storage budget is unknown, such comparison is done at different s.

in which only a subset of parameters are used to compute cardinality and thus the complexity is at most linear to s).

Building cost. As illustrated in Table 2, prior workload-driven CE solutions [8, 15] train models upon the incoming workload and predict cardinality given a query predicate. The model training incurs an overhead which can be small but also depends on adequate query predicates from the training workload; the query predicates in later-on tests should remain the same distribution - when there is a drift, the training set needs to be re-collected and the model needs to be re-built. On the other hand, prior data-driven CE models have to be built offline with multiple passes on the data and the training cost ranges from tens to hundreds of minutes *per-dataset* as reported in [21, 11]. In comparison, building samples incurs a one-pass scan or can be done during data injection; in terms of the building cost for CE, sampling is often much cheaper than ML-based solutions and is agnostic to the test workload. Indeed, data changes (i.e., appending and changing rows and columns) are a common problem. Both learning and sample-based solutions need to update properly which is an open question to the literature.

Which CE solution is better? We have discussed individual metrics that can be critical for CE, with both accu-

Method@storage s	Accuracy	Query answering cost	Building cost
Sampling	If y known: `upper_bound`(y, s) from Sec. 2 If y unknown, use synthetic $\hat{y}$: `upper_bound`($\hat{y}$, s) from Sec. 2	At most linear to s.	During data injection or one pass over data
Workload models	If y known: `eval`(M_s, y)	If dense model: Polynomial to s*	When adequate y arrives, model training + adaptation when y drifts
Data models	If y unknown, use synthetic $\hat{y}$: `eval`(M_s, $\hat{y}$)	If sparse model: At most linear to s	Multiple passes on data + model training.

Table 2: Qualitative comparisons of sampling and ML-based CE on test queries y or $\hat{y}$ and storage s. *: efforts such as SIMD can speed up this cost. Both accuracy and query answering cost are functions of s.

racy and query answering latency depending on the storage used. Figure 2 shows different cases to choose a better CE solution based on if a specific storage budget s and if the test workload y are known:

- At a specific storage budget s, we compare individual metrics in Table 2 on the given y or synthetic $\hat{y}$ depending on if the test workload y is known.
- With a flexible storage budget, ML-based CE must provide a valid tradeoff among the metrics-of-interest at different storage choices. We run the comparisons in Table 2 on the given y or synthetic $\hat{y}$ at different s and compute the metric crossovers.

There are two combinations of metrics that can be interesting to a database practitioner: accuracy + query cost, and the same plus building cost[2]. To decide which solution is better, the user or downstream application has to determine the weights on different metrics and how to aggregate. For example, an approximate solution in time-critical scenarios can weigh more on the building and query latency.

We note that among existing workload-driven models, LM [8] and MSCN [15] provide good accuracy and query answering latency. However, there is additional building overhead for collecting the training set, training the model, and handling workload drifts. As exemplary data-driven ML models, Naru [21] and DeepDB [11] demonstrate one accuracy/latency profile on a single test workload; their accuracy-latency tradeoff is unclear at different storage choices. Since theoretical error bounds for ML-based CE often do not exist, we have to rely on empirical evaluations and we refer to [20] for more detailed analyses.

So far, ML-based solutions have not shown Pareto improvements over random sampling in accuracy, building and query answering costs simultaneously. For data-driven CE solutions in which a known y does not exist, choosing synthetic $\hat{y}$ to evaluate will be discussed later.

Is ML a silver bullet to CE? Beyond the metrics of interest discussed above, learned database components devote their efforts to fit to the training set and hope for unchanged test distributions. Indeed, when this is the case, ML can provide good accuracy (i.e., `eval`() in Table 2 yields low errors). However, in practice a fixed and known test workload is rare – otherwise, we may simply memorize the workload and upon that various data structures can be built to speed up querying the workload history and to perform proper interpolation if necessary.

[2]Other combinations or considering only single metrics are invalid; e,g., for accuracy+building cost, use a sample as large as the original dataset; for only query answering latency, give a constant answer regardless of the predicate.

For example, if we have a workload history with {predicate, cardinality} pairs and such workload does not change at all, we may simply look up the workload history to answer exact cardinality if the test predicate has been seen. A better way is to build LSH [6] or a k-d tree on the predicates with leaf nodes storing the true cardinality; doing so is faster than linear lookups and also provides exact answers. Prior workload-driven models [8, 15] *generate* training and testing predicates using the same, pre-defined procedure, which is unlikely to happen in practice; real test workloads can never be generated by pre-defined code – otherwise, again, we may exploit such code and memorize the results.

One can argue that ML models learn how to better inter- or extrapolate for *unseen* inputs. This is probably true, especially when the training and test workloads follow some explicit distributions (e.g., linear, normal, zipf, functional dependency). In practice, this is not guaranteed if we do not know the test predicates; moreover, ML models are lossy and often cannot achieve perfect overfit on the training set just because real data seldom follows any presumed distribution. Even if it does, we may apply the same interpolation in the search and data structure mentioned above. Strict functional dependency (e.g., Col Y = Col X + 1) also rarely happens and many schema design guidelines suggest avoiding such columns. Correlation mining solutions such as CORDs [13] also detects and eliminates functional dependencies, either *strict* or *soft*, to reduce the compute and storage costs for downstream tasks.

To this end, we are not trying to discourage ML; great opportunities remain when it is difficult or costly to store, search, and manipulate the input space. For the problem of single-table CE discussed in this paper, the input space is yet simple; applying existing ML models seems to have only marginal benefits but at the cost of accuracy loss and compute overhead. We consider an ML-based CE to be good only when it provides good accuracy, low building and query answering costs at the same time, as well as operates at different storage budget choices so that a valid tradeoff among the metrics-of-interest can be made and chosen by a downstream application.

Evaluating workload-agnostic CE solutions. CE solutions such as sampling and data-driven models [21, 11] are not trained with a prior workload and can be queried with any test predicates. Deciding a fair test workload is critical for accuracy evaluations. As shown in our bound and Table 2, sampling already provides a tight upper error bound when storage is high or when true cardinality is high; that said, sampling may win easily in these scenarios. To construct a synthetic workload $\hat{y}$ for better accuracy evaluations, both test predicates stratified by true cardinality and a large variety of test distributions are necessary. Prior