A Learning Augmented approach to Cardinality Estimation

Mărcuș Alexandru Marian

December 17, 2025

Abstract

Cardinality estimation is the problem of determining the number distinct elements in a data stream, useful for many tasks in database systems and network monitoring. This paper presents a deep learning approach to extracting the cardinality estimate from HyperLogLog (HLL) sketches. By treating the HLL sketch as a feature vector for a regression model, the learned estimator can outperform classical bias-correction techniques in specific regimes while maintaining the space complexity of the original HLL sketch.

1 Introduction

Determining the number of distinct elements in a data stream is an active research area, useful in a wide range of fields in computer science. While the cardinality can be computed with trivial methods, for many applications, this is impractical as it would require too much time and memory. For tasks where only an approximation of the cardinality is needed, there have been developed algorithms that use far less resources. Some applications of such algorithms are database systems, using them for query processing and optimization, network security monitoring and search engines [1]. A big

While HLL is theoretically elegant, its practical implementation is fraught with challenges. The standard asymptotic error formula ($\frac{1.04}{\sqrt{m}}$) fails for small cardinalities, necessitating a switch to "Linear Counting." Furthermore, in the transition zones—where the cardinality is neither small enough for Linear Counting nor large enough for the asymptotic formula—HLL suffers from significant bias.Current state-of-the-art implementations, such as Google's HLL++, attempt to mitigate this by storing large, pre-computed lookup tables and sparse arrays for bias correction. These heuristics are complex to implement, static, and derived under specific assumptions about the hash function's distribution. This raises a compelling question: Can a machine learning model learn the true mapping from sketch states to cardinality directly from data, bypassing the need for human-derived heuristics?

2 Related work

3 Methodology

3.1 Experimental Setup

To measure the performance of this method, we generated a synthetic dataset containing 200000 different HyperLogLog sketches, each sketch having a cardinality in the range of $[10, 10^8]$, with a log-uniform distribution. Assuming a uniform hash function, we can generate uniformly distributed 64 bit integers, simulating the hash values of random elements. For real-world datasets we are using Openadresses and NCVoter. [1]

3.2 Model matematic

Algorithm 1 Learned HyperLogLog

Require: Let $h : \mathcal{D} \to \{0,1\}^{64}$ hash data from domain $\mathcal{D}$. Let $m = 2^p$ with $p \in [4..16]$.

Phase 0: Initialization.

Initialize m registers $M[0]$ to $M[m-1]$ to 0.

Phase 1: Aggregation.

for all $v \in S$ **do**

  $x := h(v)$

  $id := (x_{63}, \dots, x_{64-p})_2$ ▷ First p bits of x

  $w := (x_{63-p}, \dots, x_0)_2$

  $M[id] := \max(M[id], \rho(w))$

end for

Phase 2: Result computation.

return Model Prediction

Instead of using the classic formula $E := \alpha_m m^2 \left(\sum_{j=0}^{m-1} 2^{-M[j]}\right)^{-1}$ [2] for extracting the result from our sketch, we propose training a neural network to predict the cardinality. This approach should not have much overhead, in terms of memory or speed.

The model takes the raw register array $M \in \{0, 1, ..., q\}^m$ as input and outputs the prediction $\log(N)$.

Each HLL sketch consists of $m = 2^p$ registers, $M = [M_1, M_2, ..., M_m]$, $M_i \in \{0, 1, ..., q\}$

Before feeding the array M to the network, we normalize the values $x_i = M_i/q$. We apply three 1D convolutional layers, to downsample from the 2^p channels down to a compact 64x1024 activation map. Then it is fed into a multilayer perceptron, with three linear layers, from 64*1024 -¿ 512 -¿ 128 -¿ final prediction, $\log(n)$.

As the loss function, we chose the mean squared logarithmic error

$$L = \frac{1}{n} \sum_{i=0}^{n} \Big(\log(1 + \hat{N}) - \log(1 + N) \Big)$$

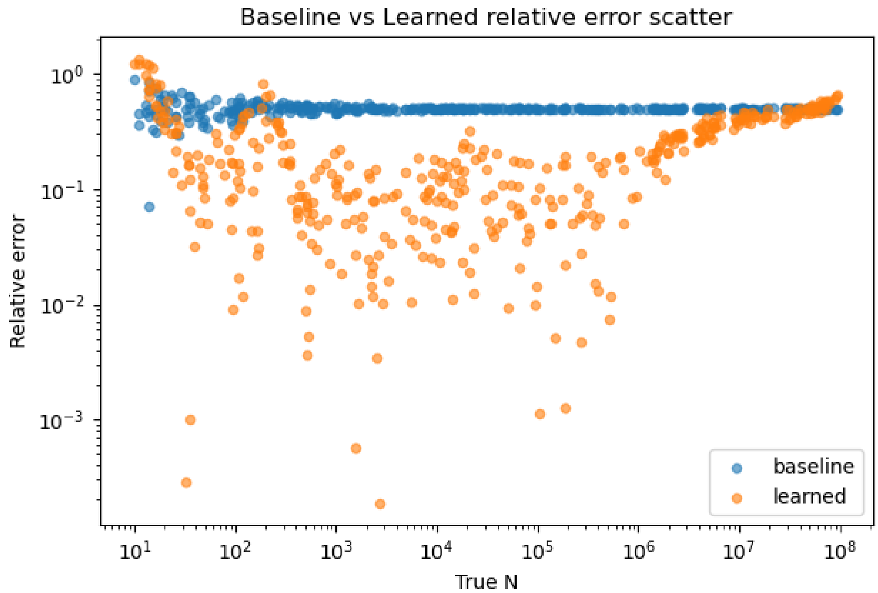

Figure 1: Our Approach (orange) vs Original HLL (blue)

3.3 Results Validation

For validating our results we are plotting the relative error $error := \frac{|\hat{N}-N|}{N}$. In the literature, the "raw" HLL estimate shows different performances for different ranges of cardinality, that are often empirically determined and corrected. It's important we observe the performance of our approach for all the possible different cardinalities, thus we prefer plotting our results instead of another measure like the mean relative error.

4 Studiu de caz

On our synthetic dataset we obtain better results than the original HLL paper, as shown in our comparison graph1. We still have to measure the overhead produces by this approach, even if the theoretical space and time complexity remains the same.

5 Related work

We compare our results to the classic HLL [2], HyperLogLog++ [3] and other learned methods [4] [5]. The ideal result would be an error close to the theoretical limit $1.04/sqrt(m)$.

Our approach builds the HyperLogLog sketch like usual, as proposed by [2], but extracts the cardinality information using a machine learning approach. This is in contrast to other machine learning approaches that try to sample from the dataset to estimate its cardinality. Building the sketch requires a scan of the dataset so it is better suited for streaming environments. A more comprehensive review is found in [1].

The advantages of this approach are reduced bias for real world distribution, because the hash function may not be able to produce fully uniform results, but the network could learn this and not be affected. It also eliminates the need for magic numbers and special cases, used in the original algorithm and other improvements [3].

References

[1] Hazar Harmouch and Felix Naumann. Cardinality estimation: an experimental survey. *Proc. VLDB Endow.*, 11(4):499–512, December 2017.

[2] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete Mathematics & Theoretical Computer Science*, DMTCS Proceedings vol. AH, 2007 Conference on Analysis of Algorithms (AofA 07), Jan 2007.

[3] Stefan Heule, Marc Nunkesser, and Alexander Hall. Hyperloglog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, page 683–692, New York, NY, USA, 2013. Association for Computing Machinery.

[4] Aiyou Chen, Jin Cao, Larry Shepp, and Tuan Nguyen. Distinct counting with a self-learning bitmap. *Journal of American Statistical Association*, 106:879–890, 2011.

[5] Renzhi Wu, Bolin Ding, Xu Chu, Zhewei Wei, Xiening Dai, Tao Guan, and Jingren Zhou. Learning to be a statistician: learned estimator for number of distinct values. *Proceedings of the VLDB Endowment*, 15(2):272–284, October 2021.

[6] Xiaoying Wang, Changbo Qu, Weiyuan Wu, Jiannan Wang, and Qingqing Zhou. Are we ready for learned cardinality estimation? *CoRR*, abs/2012.06743, 2020.

[7] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. *CoRR*, abs/1712.01208, 2017.

[8] Zhenwei Dai and Anshumali Shrivastava. Adaptive learned bloom filter (ada-bf): Efficient utilization of the classifier. *CoRR*, abs/1910.09131, 2019.

[9] Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *CoRR*, abs/2006.09123, 2020.

[10] Michael Mitzenmacher. A model for learned bloom filters and related structures. *CoRR*, abs/1802.00884, 2018.

[11] Kenneth G. Paterson and Mathilde Raynal. HyperLogLog: Exponentially bad in adversarial settings. Cryptology ePrint Archive, Paper 2021/1139, 2021.

[12] Qingzhi Ma, Ali Mohammadi Shanghooshabad, Mehrdad Almasi, Meghdad Kurmanji, and P. Triantafillou. Learned approximate query processing: Make it light, accurate and fast. In *Conference on Innovative Data Systems Research*, 2021.

[13] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *CoRR*, abs/1802.05399, 2018.

[14] Keerti Anand, Rong Ge, and Debmalya Panigrahi. Customizing ml predictions for online algorithms, 2022.

[15] Otmar Ertl. New cardinality estimation algorithms for hyperloglog sketches. *CoRR*, abs/1702.01284, 2017.

[16] Kangfei Zhao, {Jeffrey Xu} Yu, Hao Zhang, Qiyan Li, and Yu Rong. A learned sketch for subgraph counting. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 2142–2155, 2021. Publisher Copyright: © 2021 ACM.; 2021 International Conference on Management of Data, SIGMOD 2021 ; Conference date: 20-06-2021 Through 25-06-2021.

[17] Brian Tsan, Asoke Datta, Yesdaulet Izenov, and Florin Rusu. Approximate sketches. *Proc. ACM Manag. Data*, 2(1), March 2024.