Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 07, July-2023

Harnessing Learning for Database Optimization: A Paradigm Shift

Ujjwal Gupta
Senior software engineer, Walmart Global Tech

***Abstract*— Learning has emerged as a powerful tool in various domains, revolutionizing traditional approaches and achieving remarkable results. In the realm of database management and optimization, researchers and practitioners have started exploring the potential of learning techniques to enhance performance and efficiency. This paper presents a comprehensive review of learning-based database optimizations, highlighting recent advancements and their impact on various aspects of database systems. In this paper, we explore the initial endeavors in this field, examine recent progress, identify constraints and unresolved aspects, and propose potential avenues for future research.**

Keywords- Learning, deep learning, optimization

I. INTRODUCTION

For over four decades, query optimization has remained a dynamic and extensively researched domain. However, the problem's combinatorial complexity has posed challenges in devising a single comprehensive solution. As a result, the current research landscape predominantly revolves around heuristic and cost-based approaches, which may yield satisfactory results under certain limiting assumptions. Numerous techniques, such as pruning and randomized methods, have been proposed to address query optimization. However, in certain edge cases, these techniques tend to falter and generate suboptimal execution plans.

The emergence of machine learning and deep learning technologies has sparked a new trend in data management research. This trend explores the potential benefits of augmenting existing programmed heuristics with learned approaches, employing dynamic, self-learning methods for cost modeling, and enhancing traditional plan generation strategies through learning from past planning instances. These advancements aim to significantly reduce search time for future planning tasks.

In this realm, deep neural network-based learning techniques are widely utilized, falling into two broad categories: query-based and data-based approaches. Query-based methods are typically supervised learning models, trained on running queries and utilizing statistics, such as cardinalities, as labels. On the other hand, data-based methods employ unsupervised learning to capture distributions and correlations within the joint probability density functions of the underlying data.

By incorporating machine learning and deep learning into data management, researchers envision optimizing various aspects of the process, making it more efficient, adaptable, and capable of learning from its own experiences. These developments hold promise for revolutionizing data management systems and driving significant advancements in the field.

II. REVIEW SCOPE

This review article strives to offer an extensive examination of optimization methodologies achieved via learning or deep learning, focusing on (a) the progress made in learning query optimization, the knowledge acquired, and the existing gaps; (b) the comparative advantages of learning-based approaches over or in conjunction with traditional query optimization; and (c) the potential impact of learning query optimization on production systems. This paper plans to discuss the areas listed below where significant research has happened in the area of optimizations in databases with learning.

1. Query optimization with learning
2. Cost estimation with learning
3. Runtime prediction with learning
4. Benchmarks, metrics, and data preparation
5. Future research scope

III. QUERY OPTIMIZATION WITH LEARNING

a. On system design level, some experts believe that ML can pave the way for smarter and more user-friendly DBMSs, making tasks like maintenance, tuning, and optimization easier. There is even a proposition that learned components could entirely replace fundamental elements of a DBMS, including data structures, indices, sorting algorithms, and query execution processes [6], [19], [48]. Several system frameworks, such as SageDB [18] and openGauss [24], are actively exploring this direction.

SageDB[18] uses ML algorithms to learn from historical query performance data. By analyzing past query execution patterns, it can identify common bottlenecks and inefficiencies. With this knowledge, SageDB can automatically suggest better query plans and optimize the execution process, leading to improved overall performance.
openGauss[24] employs advanced query optimizers that analyze SQL queries to find the most efficient execution plans. It considers factors like available indexes, data distribution, statistics, and resource usage to generate optimal query execution plans, resulting in faster query response times.

b. Query optimizations are attempted with join ordering too. Reinforcement learning (RL) has found application in optimizing join ordering in the context of query execution. Two prominent examples are DQ [20] and ReJOIN [32], which leverage RL in combination with a pre-defined cost
model. This combination allows them to autonomously learn effective search strategies for exploring the vast space of potential join orderings.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 07, July-2023

RTOS [22] builds upon DQ and ReJOIN by introducing a Tree-LSTM, a specialized type of recurrent neural network, to calculate query costs. By using this enhanced cost estimation, RTOS further refines the join ordering process.
SkinnerDB [45] takes a different approach by employing RL to learn the best join orders in real-time during query execution. This approach enables quick switching between different join orders, leading to more efficient query processing.
AlphaJoin [23] adopts the Monte Carlo Tree Search (MCTS) technique to determine join ordering. Additionally, it employs a decision network (ADN) to choose between plans generated by AlphaJoin for longer queries and those produced by PostgreSQL for shorter queries.

c. Optimizers have also been explored in this area for improving query optimization with learning. Neo [33]: Taking a more radical approach, Neo replaces many traditional optimizer components with ML models and deep neural networks. This integration of advanced ML techniques allows Neo to optimize queries effectively, catering to a variety of scenarios.
Bao [31]: Bao adopts a schema and data agnostic approach to learning optimization. It executes queries with predefined hints and maintains a ranked plan using a tree convolutional neural network (TCNN). Periodic retraining ensures a balance between exploring new optimization strategies and exploiting learned knowledge.
Elfino [4] is a self-driving query optimizer that monitors queries throughout their entire lifecycle. It leverages an AI algorithm to learn from the mistakes made by the Spark Optimizer, enabling continuous improvement and more efficient query optimization.
Microlearner [14]: Focusing on complex cloud workloads, Microlearner divides the workload into subsets and learns micro-models independently and in parallel. This decentralized learning approach allows Microlearner to optimize query performance efficiently.

IV. COST ESTIMATION WITH LEARNING

a. Selectivity estimation is a fundamental aspect of query optimization. Some research works in this area are as follows: LATEST [37] adopts a combination of classification and decision trees to determine the most suitable selectivity estimators for time windows in streaming data scenarios. This approach enables LATEST to dynamically choose the best estimator for each time window, leading to more accurate selectivity predictions.

ASTRID [39] takes a different approach by building neural language models specifically for selectivity estimation of prefix and substring queries. By employing these language models, ASTRID can better predict the selectivity of such queries, leading to improved query optimization.

A. Dutt [7] proposes lightweight models, such as neural networks and tree-based ensembles, offer a more flexible and

adaptive approach to selectivity estimation.

[11] demonstrates improved accuracy in predicting query selectivity by using deep learning models for selectivity estimation of multi-attribute queries. The authors utilize neural density estimation to construct a selectivity estimator for the joint probability distribution based on data samples.

b. Cardinality. R. Hayek and O. Shmueli[12] uses CRN (Cardinality Estimation using Containment Rates) technique that employs deep learning to estimate cardinalities by analyzing the containment rates among queries.

Incorporating uncertainty information into a deep learning model, Fauce [25] enhances cardinality estimation accuracy. This approach utilizes deep ensembles to capture diverse patterns and provides not only cardinality estimates but also confidence intervals, resulting in a more robust estimation process.

DeepDB [13] introduces a data-driven approach for database management that is capable of accommodating workload and data changes. It enables fast retraining, ensuring adaptability to evolving conditions. Moreover, it maintains reasonable accuracy even for queries not previously encountered, making it a versatile and efficient solution.

In [36], a comprehensive evaluation of deep learning techniques is conducted, focusing on the trade-offs between model size, training time, and prediction accuracy.

UAE (Uncertainty-Aware Estimation) [42] is a cardinality estimation technique that utilizes both data and queries to achieve accurate cardinality estimates, even in scenarios where there are shifts in the data or workload. This approach is designed to handle dynamic and evolving database environments effectively, ensuring reliable cardinality estimation despite changes in the data or query patterns.

c. Cost model estimation: In [29], the feature space is partitioned using decision trees, and for each partition or split, a regression model is built.

CherryPick [2] constructs performance models by identifying deployments that meet a specified performance target through the use of Bayesian optimization. CLEO [40] adopts a learning-based approach to construct cost models from production workloads. These cost models are then integrated into a query optimizer.

V. RUNTIME PREDICTION WITH LEARNING

In the early stages, learning techniques were employed for runtime prediction, as demonstrated in [8], [10], and [46]. However, [1] reveals that linear regression for mapping PostgreSQL's estimated execution time to actual time is ineffective. [43] proposes that optimizer estimates can be valuable when the cost model is fine-tuned just before making the estimate. Additionally, Q-Cop [44] and ActiveSLA [35] leverage query runtime prediction for admission control purposes.

Recent advancements in learning-based techniques have shown promising outcomes in runtime prediction for database management systems (DBMS). These approaches treat the DBMS as a black-box and focus on constructing efficient query runtime prediction models, as evidenced by studies such as [26], [34], and [17]. For instance, Prestroid [15] utilizes tree-structured convolutional neural networks (TCNN) to predict resource consumption in large-scale queries, reducing cost.

Furthermore, RouLette [41] employs reinforcement learning (RL) to identify shareable work among Select-Project-Join (SPJ)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 07, July-2023

queries, enabling multi-query optimization and runtime adaptation. MB2 [28] leverages machine learning to create and maintain prediction models for self-driving databases, enhancing their autonomy and efficiency.

Seagull [38], on the other hand, leverages runtime prediction to optimize resource allocation on cloud platforms, ensuring effective utilization and performance. These research endeavors demonstrate the potential of learning-based techniques in revolutionizing runtime prediction for DBMS, leading to more efficient, adaptive, and self-optimizing database systems.

VI. BENCHMARKS, METRICS, AND DATA PREPARATION

a. Benchmarks: Traditional benchmarks such as the TPC suite are not well-suited for learning-based approaches. Instead, recent benchmarks have been designed to address specific challenges, like join ordering (JOB [21]) and learned indexes [30]. However, there remains a need for more comprehensive benchmarks that are based on realistic datasets. Synthetic data, which is either too random or too easily predictable, is not sufficient for evaluating the complexities of learning-based methods.

Richer benchmarks are required, which should support dynamic workloads and schemas, provide descriptive metrics about the specialization of the tested systems, be adaptable to data and schema changes, exhibit similarity between workloads and data distributions, and consider the cost associated with training and maintaining the models [5], [30]. By creating such benchmarks, researchers can better evaluate the performance, effectiveness, and general applicability of learning-based techniques in database optimization and management.

b. Metrics and statistics: Learning optimizers have the advantage of scaling beyond conventional database statistics and can access a broader range of raw information. This includes metrics related to execution containers, processes, services, and applications. By leveraging this rich pool of data, learning techniques can create profiling models, pinpoint the root cause of failures, and generate recommendations for enhancing performance and resource allocation [3], [9]. The ability to tap into diverse data sources empowers learning-based optimizers to make informed decisions, adapt to dynamic environments, and drive significant improvements in overall system efficiency and effectiveness.

c. Data preparation: DataFarm [47] adopts a user-provided set of small workload patterns to create a labeled dataset. By employing adaptive learning, it iteratively executes sample jobs to generate labels, constructing a machine learning runtime prediction model for unlabeled jobs. Similarly, Robopt [16] offers a training data generator, which creates a query workload with query runtime estimates. Using polynomial interpolation, it labels new queries based on existing acquired labels.
On the other hand, HAL [27] takes a different approach by generalizing a learned database component across various datasets and workloads. This enables HAL to apply its knowledge efficiently and effectively to different scenarios, leading to improved performance and adaptability. These techniques demonstrate the potential of learning-based approaches in generating labeled datasets, improving prediction models, and enhancing database components for a wide range of workloads and datasets.

VII. FUTURE RESEARCH SCOPE

There are several promising directions to explore in the realm of applying machine learning to database management systems:
Identifying Consistent Techniques: Given the vast array of machine learning methods available, research should focus on identifying approaches that consistently perform well across various scenarios and workloads.
Rethinking System Architecture: This involves considering new runtime environments, decomposing monolithic database systems into mini-services for instance-optimized databases, and exploring entirely new approaches instead of merely replacing traditional databases with machine learning-based solutions.
Expanding Analysis: The scope of analysis should encompass complex data types such as graphs, spatial, and streaming data, distributed or federated architectures involving multiple execution engines, and diverse optimization objectives beyond just cost.
Generalization across Database Systems: Many machine learning-based optimization models are designed for specific database systems, limiting their general applicability. Research on developing models that can generalize across different database systems is needed.
Handling Complex Query Structures: Existing research often focuses on simple queries, but many real-world queries involve complex structures. Developing machine learning models that can efficiently optimize complex queries is an important research direction.

REFERENCES

[1] M. Akdere et al., “Learning-based query performance modeling and prediction,” in ICDE, 2012
[2] O. Alipourfard et al., “Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics,” in NSDI, 2017.
[3] A. Arvanitis et al., “Automated performance management for the big data stack,” in CIDR, 2019
[4] S. Babu and A. Popescu, “Using ai to build a self-driving query optimizer,” in Spark+AI Summit, 2018.
[5] L. Bindschaedler et al., “Towards a benchmark for learned systems,” in ICDE Workshops, 2021.
[6] J. Ding et al., “ALEX: an updatable adaptive learned index,” in SIGMOD, 2020.
[7] A. Dutt et al., “Selectivity estimation for range predicates using lightweight models,” in PVLDB, 2019.
[8] A. Ganapathi et al., “Predicting multiple metrics for queries: Better decisions enabled by machine learning,” in ICDE, 2009.
[9] H. Grushka-Cohen et al., “Diversifying database activity monitoring with bandits,” CoRR:abs/1910.10777, 2019.
[10] C. Gupta et al., “Pqr: Predicting query execution times for autonomous workload management,” in ICAC, 2008.
[11] S. Hasan et al., “Deep learning models for selectivity estimation of multi-attribute queries,” in SIGMOD, 2020.
[12] R. Hayek and O. Shmueli, “Improved cardinality estimation by learning queries containment rates,” in EDBT, 2020, pp. 157–168.
[13] B. Hilprecht et al., “DeepDB: Learn from data, not from queries!” PVLDB 13(7), 2020.
[14] A. Jindal et al., “Microlearner: A fine-grained learning optimizer for big data workloads at microsoft,” in ICDE, 2021.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 07, July-2023

[15] J. K. Z. Kang et al., "Efficient deep learning pipelines for accurate cost estimations over large scale query workload," in SIGMOD, 2021.

[16] Z. Kaoudi et al., "Ml-based cross-platform query optimization," in ICDE. IEEE, 2020.

[17] X. Zhou et al., "Query performance prediction for concurrent queries using graph embedding," in PVLDB 13(9), 2020.

[18] T. Kraska et al., "Sagedb: A learned database system," in CIDR, 2019.

[19] T. Kraska et al., "The Case for Learned Index Structures," in SIGMOD, 2018.

[20] S. Krishnan et al., "Learning to optimize join queries with deep reinforcement learning," 2018, CoRR:abs/1808.03196.

[21] V. Leis et al., "How good are query optimizers, really?" in PVLDB 9(3), 2015.

[22] X. Yu et al., "Reinforcement learning with tree-lstm for join order selection," in ICDE, 2020.

[23] J. Zhang, "Alphajoin: Join order selection a la alphago," in ` PVLDB-PhD, 2020.

[24] G. Li et al., "openGauss: An autonomous database system," in PVLDB 14(12), 2021.

[25] J. Liu et al., "Fauce: Fast and accurate deep ensembles with uncertainty for cardinality estimation," PVLDB 14(11), 2021.

[26] L. Ma et al., "Query-based workload forecasting for self-driving database management systems," in SIGMOD, 2018.

[27] L. Ma et al., "Active learning for ML enhanced database systems," in SIGMOD, 2020.

[28] L. Ma et al., "MB2: decomposed behavior modeling for self-driving database management systems," in SIGMOD, 2021.

[29] T. Malik et al., "A black-box approach to query cardinality estimation," in CIDR, 2007.

[30] R. Marcus et al., "Benchmarking Learned Indexes," inPVLDB 14(1),2021.

[31] R. Marcus et al., "Bao: Making learned query optimization practical," in SIGMOD, 2021.

[32] R. Marcus and O. Papaemmanouil, "Deep reinforcement learning for join order enumeration," in aiDM@SIGMOD, 2018.

[33] R. C. Marcus et al., "Neo: A learned query optimizer," in PVLDB 12(11), 2019.

[34] R. C. Marcus and O. Papaemmanouil, "Plan-structured deep neural network models for query performance prediction," inPVLDB 12(11), 2019.

[35] P. Xiong et al., "Activesla: a profit-oriented admission control framework for database-as-a-service providers," in SoCC, 2011.

[36] J. Ortiz et al., "Learning state representations for query optimization with deep reinforcement learning," in DEEM@SIGMOD, 2018.

[37] M. Patil and A. Magdy, "LATEST: learning-assisted selectivity estimation over spatio-textual streams," in ICDE, 2021.

[38] O. Poppe et al., "Seagull: An infrastructure for load prediction and optimized resource allocation," in PVLDB 14(2), 2021.

[39] S. Shetiya et al., "Astrid: Accurate selectivity estimation for string predicates using deep learning," in PVLDB 14(4), 2021.

[40] T. Siddiqui et al., "Cost models for big data query processing: Learning, retrofitting, and our findings," in SIGMOD, 2020.

[41] P. Sioulas and A. Ailamaki, "Scalable multi-query execution using reinforcement learning" in SIGMOD, 2021

[42] P. Wu and G. Cong, "A unified deep model of learning from both data and queries for cardinality estimation," in SIGMOD, 2021.

[43] W. Wu et al., "Predicting query execution time: Are optimizer cost models really unusable?" in ICDE, 2013.

[44] S. Tozer et al., "Q-cop: Avoiding bad query mixes to minimize client timeouts under heavy loads," in ICDE, 2010.

[45] I. Trummer et al., "Skinnerdb: Regret-bounded query evaluation via reinforcement learning," in SIGMOD, 2019.

[46] S. Venkataraman et al., "Ernest: Efficient performance prediction for large-scale advanced analytics," in NSDI, 2016.

[47] F. Ventura et al., "Expand your training limits! generating training data for ml-based data management," in SIGMOD, 2021.

[48] A. Wasay et al., "Deep learning: Systems and responsibility," in SIGMOD, 2021.