Machine Learning for Data Management: A System View

Guoliang Li
Department of Computer Science
Tsinghua University, Beijing, China
liguoliang@tsinghua.edu.cn

Xuanhe Zhou
Department of Computer Science
Tsinghua University, Beijing, China
zhouxuan19@mails.tsinghua.edu.cn

***Abstract*—Machine learning techniques have been proposed to optimize data management in recent years. Compared with traditional empirical data management, learning-based methods extract knowledge from historical tasks, generalize the extracted knowledge to similar new tasks, and can achieve better performance in many scenarios (e.g., knob tuning, cardinality estimation). However, data management systems require to handle various and dynamic workloads in different scenarios, and there are some challenges in applying machine learning techniques for data management systems. First, with various workloads and hundreds of system metrics, how to select and characterize effective features for data management problems? Second, with diversified machine learning models, how to design the proper models? Third, with various data management requirements, how to validate whether the machine learning models can meet the requirements? In this tutorial, we discuss existing learning-based data management studies and how they solve the above challenges, and provide some future research directions.**

***Index Terms*—data management, machine learning**

I. Introduction

Traditional data management techniques (e.g., data partition, knob tuning, query rewrite, cost estimation) are based on empirical methodologies and specifications, and require heavy human involvement to tune and maintain the databases. However, existing empirical techniques cannot meet the high-performance requirement for various applications, large-scale database instances, and diversified users, especially on the cloud. Fortunately, machine learning techniques have been proposed to optimize data management in recent years [1], [43], [53]. Compared with traditional empirical data management, learning-based methods extract knowledge from historical tasks, generalize the extracted knowledge to similar new tasks, and can achieve better performance in many scenarios. For instance, deep learning can improve the quality of cost estimation [8], [37], [38], [41], [46], and deep reinforcement learning can be utilized to optimize various configuration settings [3], [10], [23], [42], [50], [56].

However, there are several challenges in applying machine learning for data management problems. First, it is challenging to select effective features for the machine learning models. Data management systems are very complex with various workloads and runtime characteristics (e.g., read/write ratio, buffer hit rate). It is hard to detect useful features to model them. Moreover, some features may not be available in online scenarios (e.g., query logs). Second, there are various machine learning models like supervised/unsupervised/reinforcement-learning. These models are suitable for different problems. For example, deep learning can work well to regress for high-dimension table data, while reinforcement learning is good at searching good query plans. It is challenging to design machine learning models to solve different data management problems. Third, there are various data management requirements. For example, query rewrite requires relatively low latency (e.g., milliseconds) and it is intolerable to take hours for applying reinforcement learning to select rewrite rules. Besides, it requires a unified machine learning (ML) data/model management for different learning-based components. There are many techniques to address these challenges recently and we plan to summarize these techniques.

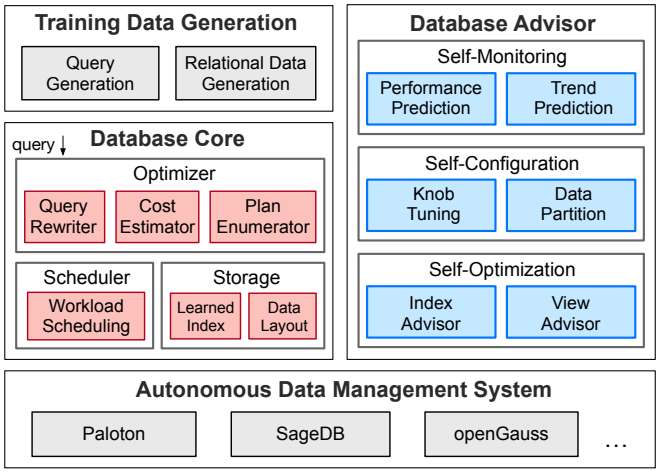

Fig. 1. Overview of Machine Learning for Data Management.

Tutorial Overview. We will provide a 3 hours tutorial to review existing ML techniques for data management [1].

(1) Background and Motivation (20min). We first introduce the background and motivation of ML for data management.

(2) Database Core (60min). We then review existing ML techniques for optimizing database core components, e.g., query rewriter [55], cost estimator [8], [37]–[39], [41], [46], plan enumerator [30], [31], [48], workload scheduling [36], and data structures like learned index [7], [14], [33], [45].

(3) Database Advisor (40min). Next we summarize database advisor techniques, including self-monitoring (performance prediction, trend prediction) [28], [32], [57], self-configuration

[1] dbgroup.cs.tsinghua.edu.cn/ligl/papers/ai4db-slides.pdf

(knob tuning, data partition) [12], [23], [52], and self-optimization (index/view advisors) [16], [20], [49], [56].

(4) Training Data Generation (20min). It is vital for learning-based methods to generate training data for model training or performance validation. Related works include query generation [51] and relational data generation [9].

(5) Autonomous Systems (20min). ML-driven data management systems are studied by both the academia and industry [17], [24], [29], [34], [54]. For example, `SageDB` [17] gives a vision to specialize the database implementation by learning the data distribution. And `openGauss` [24] provides a practical autonomous framework that validates, trains, and implements learned components on an open-sourced database.

We provide research challenges (20min), e.g., credible machine learning for data management and model adaptivity.

Target Audience. The intended audience include ICDE attendees from both research and industry communities that are interested in data management and machine learning. We will not require any prior background knowledge in machine learning. The tutorial will be self-contained.

Difference with other Tutorials. There are some tutorials on machine learning for database [21], [22], [35], [43]. Different from them, we focus on the research challenges of designing a learning-based database system from a system view.

II. Tutorial Outline

Traditional database management is based on empirical methodologies and specifications, and requires great human efforts to tune and maintain the databases. ML techniques can be used to alleviate these limitations – exploring larger design space than humans and replacing heuristics to address NP-hard problems. We categorize existing techniques as below.

Database Core. It aims to utilize machine learning techniques to address the NP-hard problems in database optimization, including query rewriter, cost estimator, plan enumerator, workload scheduling, and learned indexes.

(1) *Query Rewriter*. Query rewriter aims to optimize the redundant or inefficient operators in logical query and enhance the query execution. However, there are numerous rewrite orders for a query (e.g., different operators and applicable rules), and traditional empirical rewriting methods only rewrite in a fixed order (e.g., top down) and may derive suboptimal queries. Instead, based on human-crafted rules, deep tree searching algorithms like MCTS can judiciously select the appropriate rules and apply the rules in a good order [55].

(2) *Cost Estimator*. Database optimizer relies on estimated costs to select an optimized plan, but traditional techniques cannot effectively capture the correlations between different columns/tables and thus cannot provide high-quality estimation. Recently, learning-based techniques (e.g., CNN [8], RNN [37], Generative Model [41]) are proposed to estimate the cardinality and cost, where they utilize deep neural networks to capture data correlations. For example, an LSTM based work [37] learns a representation for each sub-plan with physical operator and predicates, and outputs the estimated cardinality and cost simultaneously with an estimation layer. Besides, there are some empirical evaluations that verify the suitable scenarios of different estimation methods [39], [44].

(3) *Plan Enumerator*. A SQL query may have billions of possible plans and it is vital to efficiently find a good plan under resource constraints. Traditional heuristics methods cannot find optimal plans for dozens of tables and dynamic programming is costly to explore the huge plan space. Thus there are some deep reinforcement learning based methods [30], [31], [48] that automatically select good plans. For example, SkinnerDB [40] proposed to optimize the join order on the fly and use a Monte-Carlo tree search based method to try out different join orders in each time slice.

(4) *Workload Scheduling*. Effective workload scheduling can greatly improve the performance by improving resource usage and avoiding data conflicts. Traditional methods either schedule workload sequentially, which cannot consider potential conflicts, or schedule workloads based on the costs estimated by the optimizer. Sheng et al. [36] proposed a learning-based transaction scheduling method, which can balance concurrency and conflict rates using supervised algorithms.

(5) *Learned indexes* [7], [14], [33], [45] are proposed for reducing the index size, improving access efficiency with the indexes, and supporting dynamic indexes. For example, Kraska et al. [18] proposed *indexes as models*, where the B^+tree index can be seen as a model that maps each query key to its page. Learned indexes are also widely studied for data updates and high-dimensional data. Besides, there are some learned data layout works [6] that judiciously store data so as to reduce disk I/O and enhance data access.

Database Advisor. It aims to utilize ML models to provide autonomous services, including performance prediction, trend prediction, knob tuner, data partition, and index/view advisors.

(1) *Performance Prediction.* Performance prediction is vital to meet the service level agreements (SLAs). Marcus et al. [32] used deep learning to predict query latency based on interactions between child/parent operators and parallel plans. To capture detailed query operator relations (e.g., data sharing/conflict), Zhou et al. [57] proposed to characterize concurrent queries into a graph model based on the execution mechanisms (e.g., shared buffer, locks) and utilize deep graph embedding to map the graph into performance metrics.

(2) *Trend Prediction.* Traditional trend prediction methods are rule-based [4], which use domain knowledge of database engines (e.g., latency, resource utilization) to identify signals relevant to workload characteristics. However, it is time-consuming to rebuild a statistics model when workload changes. So Ma [28] proposed an ML-based system that predicts the future trend of different workloads.

(3) *Knob tuning*. Databases have hundreds of tunable system knobs (e.g., `Work_Mem`, `Max_Connections`) [27], which control many important aspects of databases (e.g., memory allocation, I/O control, logging) and affect database performance. To facilitate traditional manual tuning, researchers utilize learning-based techniques [23], [50], [52] to automate knob tuning, which not only achieve higher tuning performance but less tuning time. For example, CDBTune [50]

TABLE I
METHODS OF MACHINE LEARNING FOR DATA MANAGEMENT

Category	Problem	Technique	Target	Adaptivity
Database Core	Query Rewrite	MCTS [55]	Overhead, Performance	✓ (Query&Schema Encoding)
	Cost Estimation	Kernel Density [8], [11]	Accuracy	
		Sum-Product [13]		✓ (Table Data Sampling)
		Autoregressive [47]		✓ (Join Sampling for Join Queries)
		Deep Learning [8], [15], [37]		
		Generative Model [41]		✓(Various-Type Data Encoding)
	Plan Enumerator	DRL [30], [31], [48]	Performance	✓ (Query&Schema Encoding)
		Monte Carlo Tree Search [40]	Overhead, Performance	✓ (Adaptive Planning)
	Workload Scheduling	DRL [36]	Overhead, Performance	
	Learned Indexes	Deep Learning [7], [14], [33], [45]	Overhead, Performance	
Database Advisor	Performance Prediction	Deep Learning [32]	Overhead, Performance	
		Graph Convolution Network [57]		✓ (Concurrent Query Encoding)
	Trend Prediction	Clustering [28]	Accuracy	
	Knob Tuning	Gaussian Process [2], [19]	Performance	✓(Workload Mapping)
		DRL [23], [50]		✓ (Query Feature Encoding)
		Meta-Learning [52]		✓ (Workload Profiling&Mapping)
	Data Partition	DRL [12]	Performance	✓ (Workload Mapping)
	Index Advisor	DRL [20], [56]	Overhead, Performance	
	View Advisor	DRL [10]	Performance	✓ (Attention-based View Encoding)
Training Data Generation	Query Generation	Generative Model [51]	Similarity	
	Relational Data Generation	Generative Model [9]	Similarity	

models knob tuning as a sequential decision problem and relies on reinforcement learning to improve tuning performance. QTune [23] further characterizes query features using deep learning and can achieve finer granularity tuning. However, RL-based methods cannot efficiently migrate among databases. So Zhang et al. [52] proposed to extract common workload features (e.g, SQL keywords) and transfer the pre-trained models on historical databases to the tuning model on new database based on workload similarity.

(4) *Data Partition*. Traditional methods heuristically select columns as partition keys (single column mostly) and cannot balance between load balance and access efficiency. Some work [12] also utilizes reinforcement learning models to explore different partition keys and implements a fully-connected neural network to estimate partition benefits.

(5) *Index advisor*. There are some learning-based works that automatically recommend indexes [5], [20], [56]. For example, Hai et al. [20] proposed an RL-based index selection method. They first extract candidate indexes based on empirical rules, and then design a DQN model to search promising combinations from those candidate indexes.

(6) *View advisor*. Judiciously selecting materialized views can significantly improve the query performance within an acceptable overhead, especially for cloud databases with millions of database instances and numerous queries. Thus, it calls for the view advisor, which automatically identifies the appropriate views for a given query workload [16], [26]. For example, Han et al. [10] propose a deep reinforcement learning method to estimate the benefit of different MV candidates and queries, and select MVs for dynamic workloads.

Training Data Generation. Most of existing learning-based methods are data-driven. However, it is hard to collect sufficient training data from real scenarios (e.g., for privacy issues). Hence, there are some works that automatically generate desired data, e.g., query generation [51] and relational data generation [9] using generative models.

Autonomous Data Management Systems. There are some learning-based database systems which are studied by both the academia and industry [17], [24], [25], [29], [34], [54]. For example, `SageDB` [17] provided a vision to specialize the database implementation by learning the data distribution (CDF models) and designing database components based on the knowledge, e.g., learned index, learned query scheduling. Besides, Li [24] proposed an autonomous framework that validates, trains, and implements learning-based components [10], [23], [37], [48], [57] on an open-source database.

Challenges and Open Problems. There are several challenges that utilize ML techniques for data management.

(1) *Lightweight Model Training.* Existing learning-based works require numerous training samples and take long training time. But in many scenarios we need lightweight models that can be efficiently trained with limited historical data.

(2) *Model Validation.* It is hard to evaluate whether a learned model is effective and thus a validation model is required.

(3) *Adaptability.* The adaptability is a big challenge, e.g., adapting to dynamic workloads, datasets, database instances, new hardware environments, and even other database systems.

(4) *Automatic Assembling.* It is vital to automatically assemble proper machine learning models and design an autonomous database system that meets the user requirements.

(4) *Learning for OLAP.* Traditional OLAP focuses on relational data analytics. However, in the big data era, many new data types have emerged, e.g., graph data, time-series data, spatial data, it calls for new data analytics techniques to analyze these multi-model data. Moreover, besides traditional aggregation queries, many applications require to use machine learning algorithms to enhance data analytics, e.g., image analysis. Thus it is rather challenging to integrate AI and DB techniques to provide new data analytics functionality.

(5) *Learning for OLTP.* Transaction modeling and scheduling are rather important to OLTP systems, because different transactions may have conflicts. However, it is not free to model and schedule the transactions, and it calls for more efficient

models that can instantly model and schedule the transactions in multiple cores and multiple machines.

III. BIOGRAPHY

Guoliang Li is a full professor in the Department of Computer Science, Tsinghua University. His research interests mainly include data cleaning and integration, crowdsourcing, and AI4DB, DB4AI, hybrid DB&AI. He got VLDB 2017 early research contribution award, TCDE 2014 early career award, CIKM 2017 best paper award, and best of VLDB/ICDE. He will present motivation, database core and open problems.

Xuanhe Zhou is currently a PhD student in the Department of Computer Science, Tsinghua University. His research interests lie in autonomous database systems. He will present database advisor and training data generation.

Acknowledgement This paper was supported by NSF of China (61925205, 61632016, 62102215), Huawei, TAL education, and Beijing National Research Center for Information Science and Technology (BNRist).

REFERENCES

[1] https://github.com/tsinghuadatabasegroup/aidb.
[2] D. V. Aken, A. Pavlo, and et al. Automatic database management system tuning through large-scale machine learning. In *SIGMOD*, 2017.
[3] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.
[4] S. Das, F. Li, V. R. Narasayya, and et al. Automated demand-driven resource scaling in relational database-as-a-service. In *SIGMOD*, 2016.
[5] B. Ding, S. Das, R. Marcus, and et al. AI meets AI: leveraging query executions to improve index recommendations. In *SIGMOD*, 2019.
[6] J. Ding, U. F. Minhas, B. Chandramouli, C. Wang, Y. Li, Y. Li, D. Kossmann, J. Gehrke, and T. Kraska. Instance-optimized data layouts for cloud analytics workloads. In *SIGMOD*, pages 418–431, 2021.
[7] J. Ding, U. F. Minhas, J. Yu, and et al. ALEX: an updatable adaptive learned index. In *SIGMOD*, pages 969–984, 2020.
[8] A. Dutt, C. Wang, A. Nazi, and et al. Selectivity estimation for range predicates using lightweight models. *VLDB*, 12(9):1044–1057, 2019.
[9] J. Fan, T. Liu, G. Li, and et al. Relational data synthesis using generative adversarial networks: A design space exploration. *VLDB*, 2020.
[10] Y. Han, G. Li, H. Yuan, and J. Sun. An autonomous materialized view management system with deep reinforcement learning. In *ICDE*, 2021.
[11] M. Heimel, M. Kiefer, and V. Markl. Self-tuning, gpu-accelerated kernel density models for multidimensional selectivity estimation. In T. K. Sellis, S. B. Davidson, and et al, editors, *SIGMOD*, 2015.
[12] B. Hilprecht, C. Binnig, and U. Röhm. Learning a partitioning advisor for cloud databases. In *SIGMOD*, pages 143–157, 2020.
[13] B. Hilprecht, A. Schmidt, M. Kulessa, A. Molina, and et al. Deepdb: Learn from data, not from queries! *VLDB*, 13(7):992–1005, 2020.
[14] S. Idreos, N. Dayan, and et al. Design continuums and the path toward self-designing key-value stores that know and learn. In *CIDR*, 2019.
[15] A. Kipf, T. Kipf, and et al. Learned cardinalities: Estimating correlated joins with deep learning. In *CIDR*, 2019.
[16] J. Kossmann, S. Halfpap, M. Jankrift, and R. Schlosser. Magic mirror in my hand, which is the best in the land? an experimental evaluation of index selection algorithms. *VLDB*, 13(11):2382–2395, 2020.
[17] T. Kraska, M. Alizadeh, A. Beutel, E. H. Chi, and et al. Sagedb: A learned database system. In *CIDR*, 2019.
[18] T. Kraska, A. Beutel, and E. H. C. et al. The case for learned index structures. In *SIGMOD*, pages 489–504, 2018.
[19] M. Kunjir and S. Babu. Black or white? how to develop an autotuner for memory-based analytics. In *SIGMOD*, pages 1667–1683, 2020.
[20] H. Lan, Z. Bao, and Y. Peng. An index advisor using deep reinforcement learning. In *CIKM*, pages 2105–2108, 2020.
[21] G. Li, X. Zhou, and L. Cao. AI meets database: AI4DB and DB4AI. In *SIGMOD*, pages 2859–2866, 2021.
[22] G. Li, X. Zhou, and L. Cao. Machine learning for databases. *Proc. VLDB Endow.*, 14(12):3190–3193, 2021.
[23] G. Li, X. Zhou, and et al. Qtune: A query-aware database tuning system with deep reinforcement learning. *VLDB*, 12(12):2118–2130, 2019.
[24] G. Li, X. Zhou, S. Ji, X. Yu, Y. Han, L. Jin, W. Li, T. Wang, and S. Li. opengauss: An autonomous database system. *VLDB*, 2021.
[25] G. Li, X. Zhou, and S. Li. Xuanyuan: An ai-native database. *IEEE Data Eng. Bull.*, 42(2):70–81, 2019.
[26] X. Liang, A. J. Elmore, and S. Krishnan. Opportunistic view materialization with deep reinforcement learning. *CoRR*, 2019.
[27] J. Lu, Y. Chen, and et al. Speedup your analytics: Automatic parameter tuning for databases and big data systems. *PVLDB*, 2019.
[28] L. Ma, D. V. Aken, A. Hefny, G. Mezerhane, A. Pavlo, and G. J. Gordon. Query-based workload forecasting for self-driving database management systems. In *SIGMOD*, pages 631–645, 2018.
[29] L. Ma, W. Zhang, and et al. MB2: decomposed behavior modeling for self-driving database management systems. In *SIGMOD*, 2021.
[30] R. Marcus and O. Papaemmanouil. Deep reinforcement learning for join order enumeration. In *SIGMOD*, pages 3:1–3:4, 2018.
[31] R. C. Marcus, P. Negi, H. Mao, and et al. Neo: A learned query optimizer. *PVLDB*, 2019.
[32] R. C. Marcus and O. Papaemmanouil. Plan-structured deep neural network models for query performance prediction. *VLDB*, 2019.
[33] V. Nathan, J. Ding, M. Alizadeh, and T. Kraska. Learning multi-dimensional indexes. In *SIGMOD*, pages 985–1000, 2020.
[34] A. Pavlo, G. Angulo, J. Arulraj, and et al. Self-driving database management systems. In *CIDR*, 2017.
[35] C. Ré, D. Agrawal, and et al. Machine learning and databases: The sound of things to come or a cacophony of hype? In *SIGMOD*, 2015.
[36] Y. Sheng, A. Tomasic, T. Zhang, and A. Pavlo. Scheduling OLTP transactions via learned abort prediction. In R. Bordawekar and O. Shmueli, editors, *aiDM@SIGMOD*, 2019.
[37] J. Sun and G. Li. An end-to-end learning-based cost estimator. *PVLDB*, 13(3):307–319, 2019.
[38] J. Sun, G. Li, and N. Tang. Learned cardinality estimation for similarity queries. In *SIGMOD*, pages 1745–1757, 2021.
[39] J. Sun, J. Zhang, and et al. Learned cardinality estimation: A design space exploration and a comparative evaluation. *VLDB*, 2022.
[40] I. Trummer, J. Wang, D. Maram, S. Moseley, S. Jo, and J. Antonakakis. Skinnerdb: Regret-bounded query evaluation via reinforcement learning. In *SIGMOD*, pages 1153–1170, 2019.
[41] J. Wang, C. Chai, J. Liu, and G. Li. Face: A normalizing flow based cardinality estimator. *Proc. VLDB Endow.*, 15(1), 2022.
[42] J. Wang, I. Trummer, and D. Basu. UDO: universal database optimization using reinforcement learning. *VLDB*, 14(13):3402–3414, 2021.
[43] W. Wang, M. Zhang, G. Chen, H. V. Jagadish, and et al. Database meets deep learning: Challenges and opportunities. *SIGMOD Rec.*, 2016.
[44] X. Wang, C. Qu, W. Wu, J. Wang, and Q. Zhou. Are we ready for learned cardinality estimation? *Proc. VLDB Endow.*, 14(9), 2021.
[45] J. Wu, Y. Zhang, S. Chen, Y. Chen, J. Wang, and C. Xing. Updatable learned index with precise positions. *VLDB*, 14(8):1276–1288, 2021.
[46] P. Wu and G. Cong. A unified deep model of learning from both data and queries for cardinality estimation. In *SIGMOD*, 2021.
[47] Z. Yang, E. Liang, A. Kamsetty, C. Wu, and et al. Deep unsupervised cardinality estimation. *VLDB*, 13(3):279–292, 2019.
[48] X. Yu, G. Li, C. chai, and N. Tang. Reinforcement learning with tree-lstm for join order selection. In *ICDE*, pages 1297–1308, 2020.
[49] H. Yuan, G. Li, L. Feng, and et al. Automatic view generation with deep learning and reinforcement learning. In *ICDE*, 2020.
[50] J. Zhang, Y. Liu, K. Zhou, G. Li, and et al. An end-to-end automatic cloud database tuning system using deep reinforcement learning. In *SIGMOD*, pages 415–432, 2019.
[51] L. Zhang, C. Chai, X. Zhou, and G. Li. Learnedsqlgen: Constraint-aware sql generation using reinforcement learning. In *SIGMOD*, 2022.
[52] X. Zhang, H. Wu, Z. Chang, and et al. Restune: Resource oriented tuning boosted by meta-learning for cloud databases. In *SIGMOD*, 2021.
[53] X. Zhou, C. Chai, G. Li, and J. Sun. Database meets artificial intelligence: A survey. *TKDE*, 2020.
[54] X. Zhou, L. Jin, S. Ji, and et al. Dbmind: A self-driving platform in opengauss. *Proc. VLDB Endow.*, 14(12):2743–2746, 2021.
[55] X. Zhou, G. Li, C. Chai, and J. Feng. A learned query rewrite system using monte carlo tree search. *PVLDB*, 2022.
[56] X. Zhou, L. Liu, W. Li, and et al. Autoindex: An incremental index management system for dynamic workloads. In *ICDE*, 2022.
[57] X. Zhou, J. Sun, G. Li, and et al. Query performance prediction for concurrent queries using graph embedding. *VLDB*, 2020.