Deep Query Optimization

Tin Vu
Department of Computer Science and Engineering
University of California, Riverside
tin.vu@email.ucr.edu

ABSTRACT

In recent decades, we observed the rapid growth of several big data platforms. In this context, the complexity of distributed systems make it much harder to develop rigorous cost models for query optimization problems. This paper aims to address two problems of the query optimization process: cost estimation and index selection. The cost estimation problem predicts the best execution plan by measuring the cost of alternative query plans. The index selection problem determines the most suitable indexing method with a given dataset. Both problems require the development of a complex function that measures the cost or suitability of alternatives to a specific dataset. Therefore, we employ deep learning to solve those problems due to its capability of learning complicated models. We first addressed a simple form of cost estimation problem: selectivity estimation. Our preliminary results show that our deep learning models work efficiently with the accuracy of selectivity estimation up to 97%.

CCS CONCEPTS

• **Information systems → Data management systems**;

KEYWORDS

Query Optimization; Data Indexing; Deep Learning

ACM Reference Format:
Tin Vu. 2019. Deep Query Optimization. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, Article 4, 3 pages. https://doi.org/10.1145/3299869.3300104

1 PROBLEM AND MOTIVATION

First, given a query, the cost estimation problem predicts the cost of alternative query plans in order to find the optimized one. Traditional DBMS are mostly using cost-based optimizer(CBO) [4] to address this problem. Similarly, many big data systems such as Hive or SparkSQL Catalyst also use CBO in their query optimizer. The drawback of CBO is that it is not easy to convert its abstract cost metrics to real performance metrics such as number of I/O operations, elapsed time, memory, network cost, etc. Therefore, due to the lack of a proper cost model, it is mostly used as a rule-based query optimizer instead. Recently, there are some works which aim to predict query performance using machine learning models[2, 3]. The main idea of those approaches is that they extract important features of a query such as number of join operators, number of nested subqueries, total number of selection predicates. After that, they apply different machine learning models to find the relationship between those features and performance metrics. One limitation is that the feature extraction process is not only requiring expert's knowledge, but also has a chance to miss important hidden features.

Second, the index selection problem find the best indexing method for a dataset in order to optimize index performance: indexing time, index quality. In general, database administrators manually choose the indexing method based on characteristics of dataset's distribution. For example, a hashtable index should be the most suitable index if the given dataset has an uniform distribution of keys. A bitmap index is better than B-tree, hashtables if the dataset is highly compressed. Overall, the manual approach is inefficient since it requires much of human effort to extract data features from the given dataset, and select the suitable index based on these features.

The common disadvantage of current approaches for query optimization problems is that they need human effort to get the data characteristics. On the contrary, the recent success of deep learning [6] shows that it can learn the data insights without an explicit guide from human. Thus, this brings up an interesting question of whether we can apply deep learning techniques to address these problems? There are some advantages of deep learning that may address the drawbacks of previous works. First, deep learning models can be constructed by many hidden layers of nonlinear processing units for feature extraction and transformation, which allows us to learn the behavior of data without feature selection process. Second, deep learning probably allows a better representation of data since we will not miss data hidden features.

ACM ISBN 978-1-4503-5643-5/19/06.
https://doi.org/10.1145/3299869.3300104

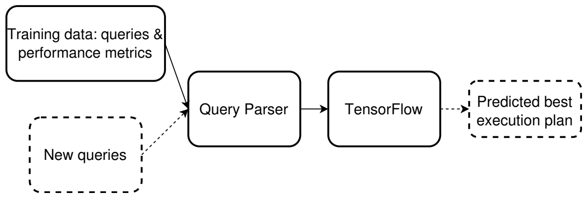
Training data: queries & performance metrics
New queries
Query Parser
TensorFlow
Predicted best execution plan

Figure 1: Query optimizer using deep learning

2 BACKGROUND AND RELATED WORK

There are some previous works which aim to use machine learning models to predict query performance. PQR[3] employs machine learning to predict execution time for changing query workload. However, this approach only focused on the query elapsed time. To address this problem, the paper[2] tried to build cost models with multiple performance metrics. In particular, they use vectorized queries and performance metrics as training data points for their models. This approach still has some drawbacks. First, it require a manual feature extraction process. Second, it did not take the dataset histogram into account, which also affects to the query's performance. Paper [7] takes database properties into account for cardinality estimation problem. However, our approach aims to go further to find optimal query plan.

For the index selection problem, the paper [5] aims to completely replace indexing component in database by machine learning models. This approach is promising but it is difficult to apply in reality since the current database systems has decades of development, which is hard to completely replace any component. In the other hand, the paper [1] proposes a decision tree for data indexing based on skewness of the data. However, this approach still need some initial parameters which is determined by human. On the contrary, our approach aims to use deep learning as a black box to guide current database systems optimize the data index component using popular indexing techniques.

3 APPROACH AND UNIQUENESS

Given a SQL query, traditional DBMS employ cost-based optimizer(CBO) [4] to determine the most efficient execution plan. However, there are many challenges to apply CBO in the big data platforms. For example, the distributed environment make it difficult to build a cost model with many aspects: I/O cost, network cost, CPU and memory cost of each node. To address the cost estimation problem, we build a deep learning model for query optimizer as shown in Figure 1. The training data points includes query and performance metrics (time, memory, I/O cost). The query is parsed to a tree and the performance metrics will be the output of a regression problem. Finally, we can use the trained model to predict best execution plan. Using deep learning for query

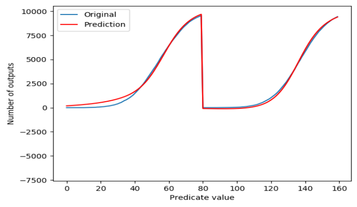

Figure 2: Predict number of query outputs using deep learning

optimization also raise some challenges. For example, one of the major limitations in deep learning is that both the input and output are fixed size and unstructured. Thus it will be challenging to use with the variable size and structured query plans as data points.

The architecture of index selection system also follows the concept of Figure 1. However, the training data points is the dataset and the indexing performance (indexing time, index quality) of different indexes (B-tree, hashtable index, bitmap index, to name a few). Given a dataset, we construct its training data points as follows: first, we compute the histogram matrix of the dataset, then convert it to a vector to feed to a deep neural networks. Second, we use the sampling-based indexing method [8] to compute the performance metrics of different indexes. The training data points are feed to a classification model, which predict the best index in terms of indexing performance. In the end, we will have a model which is able to predict the suitable index for any new given dataset.

4 RESULTS AND CONTRIBUTIONS

Figure 2 shows a preliminary result of our query optimizer using deep learning. We first solve a simple form of cost estimation problem: selectivity estimation. Given a dataset of two tables with two different Gaussian distributions of a column values and a SELECT query with a specific predicate for that column, how can we predict number of output records for this query? To solve this problem, we constructed training data points by executing SELECT queries with different predicate values, then use the trained model to predict number of output records for new queries. The results indicates that we can predict the number of outputs for a query with high accuracy of 97%. Since the number of outputs has a linear relation with query response time or other performance metrics, we can further predict the query performance using this model.

ACKNOWLEDGEMENT

This work is supported in part by the National Science Foundation (NSF) under grant IIS-1838222.

REFERENCES

[1] Alberto Belussi, Sara Migliorini, and Ahmed Eldawy. 2018. Detecting skewness of big spatial data in SpatialHadoop. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 432–435.

[2] Archana Ganapathi, Harumi Kuno, Umeshwar Dayal, Janet L Wiener, Armando Fox, Michael Jordan, and David Patterson. 2009. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*. IEEE, 592–603.

[3] Chetan Gupta, Abhay Mehta, and Umeshwar Dayal. 2008. PQR: Predicting query execution times for autonomous workload management. In *Autonomic Computing, 2008. ICAC'08. International Conference on*. IEEE, 13–22.

[4] Yannis E Ioannidis. 1996. Query optimization. *ACM Computing Surveys (CSUR)* 28, 1 (1996), 121–123.

[5] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*. ACM, 489–504.

[6] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[7] Jennifer Ortiz, Magdalena Balazinska, Johannes Gehrke, and S Sathiya Keerthi. 2018. Learning State Representations for Query Optimization with Deep Reinforcement Learning. *arXiv preprint arXiv:1803.08604* (2018).

[8] Tin Vu and Ahmed Eldawy. 2018. R-Grove: growing a family of R-trees in the big-data forest. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 532–535.