ALMSS: Automatic Learned Index Model Selection System

Rui Zhu[1], Hongzhi Wang[1,2](✉), Yafeng Tang[1], and Bo Xu[1]

[1] Harbin Institute of Technology, Harbin, China
[2] Peng Cheng Laboratory, Shenzhen, China
{20S003048,wangzh,1190201313,1190201620}@hit.edu.cn

Abstract. Index is an indispensable part of database. As we enter the era of big data, the traditional index structure is found not to support large-scale data well. Although many index structures such as learned indexes based on machine learning have been proposed to solve such problems of traditional indexes, it is a great challenge to select the most suitable learned indexes for the specific application. To solve this problem, we design ALMSS, an automatic learned index model selection system, which provides a user-friendly interface and can help users automatically select the learned index model. In this paper, we introduce the overall architecture and main technologies of ALMSS, and show the demonstration of this system.

Keywords: Learned index · Model selection · Machine learning

1 Introduction

Most of the existing database systems adopt traditional index structures, such as B+ trees. However, for big data, the traditional index structures have exposed some shortcomings. For example, B+ tree may cost too much space. At this time, the learned index was put forward. In 2018, Kraska et al. [1] first proposed the concept of combining machine learning with traditional index structures. Compared with traditional indexes, learned index can reduce the cost of index space and improve index query performance. In the following years, scientists have successively proposed many learned indexes to solve the problems of the initial version of learned index that cannot support insertion.

At present, most researches on one-dimensional learned indexes focus on data partition and data insertion strategy. For example, AIDEL [2] and ALEX [3] use the local insertion strategy, while PGM-index [4], XIndex [5] and FITing-Tree [6] use the remote insertion strategy. Local insertion strategy is to reserve a certain gap in the sorting array of leaf nodes. When some data needs to be inserted, we insert the data into gaps by using some policies. We update the model when gaps are reduced to a certain number. For example, ALEX predicts the location of data insertion through a model. If there is a gap in the predicted location, then the data can be inserted into the gap. On the contrary, if the location predicted by the model is not a gap, ALEX uses an exponential search method to search a gap and inserts the data into the gap. And the remote insertion strategy inserts the data into the buffer, and then formulates the strategy to merge the data in the buffer with the data corresponding to the original model. For example, XIndex sets a two-phase compacting strategy to support data merging. Their leaf node design strategy is relatively simple, mostly using linear regression model. However, in many cases, simple linear regression models cannot fit the data well. Therefore, how to choose the leaf node model has become an urgent problem to be solved.

In order to solve the above problems, we first designed an automatic learned index model selection system(ALMSS), which can automatically select the model according to the data set selected by users and provide a user-friendly interface. The main functions of the system are as follows:

– Automatic Index Selection: Our system implements the automatic index selection function, including not only the selection of traditional indexes such as B-tree, hash tree, etc., but also the selection of learned indexes for the automatic selection model that we design. Users can choose to use traditional indexes, such as hash, B-tree, or learning index according to their own needs. If the users choose learning index, the node model of learning index is automatically selected by our system. Users do not need to select the internal model of learning index.
– Friendly interface: ALMSS provides a graphical interface, and users can accomplish the automatic selection function of the index by simple operations such as selecting data sets and inputting SQL query statements. Users only need simple operations to achieve the required functions. At the same time, our ALMSS system will provide users with system information as much as possible, so that users can understand the internal implementation steps in our ALMSS more clearly.

Our paper is structured as follows. In Sect. 2, we will introduce the overall architecture of our ALMSS system in detail. Then in Sect. 3, based on the understanding of the overall architecture, we will introduce the key technologies of ALMSS system in more detail. Finally, in Sect. 4, we will show the demonstration scenarios of the system.

2 System Overview

Figure 1 shows the architecture of ALMSS. Users can select the appropriate dataset for the following operation, and then enter the SQL statement of the query. In the learned index module, according to the results of the automatic selection model, users can obtain accurate query location and model information involved in the query process. The learning index module includes traditional index models, such as b-tree, and machine learning models such as RMI. For example, in the upload dataset module, when the user selects the dataset from the existing dataset in the system, the corresponding dataset information will be displayed, including data set content, data distribution, etc. In SQL parsing module, users input SQL statements that need to be executed. ALMSS parses this statement regularly, and gives a large range of key location description of related fields according to uploading dataset. This description will be transmitted to the learning index module for further processing. This part provides some models or algorithms for users to reduce the scope of the key at the maximum speed. In addition, we show the training time and other information through visualization. The Automatic Selection module receives the results of the previous module, automatically selects the most suitable regression model to analyze and obtain accurate query location, and provides the selected model and its parameters to users (Fig. 1).

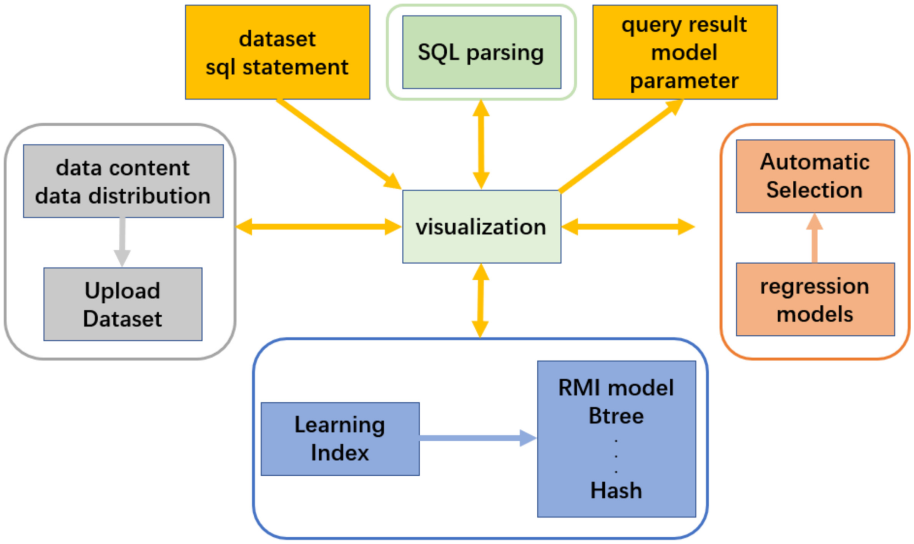


Fig. 1. System architecture

3 Key Technologies

In this section, we will introduce the key technologies of ALMSS from two parts. The core module of ALMSS is to select the best index suitable for the dataset for users. We divide the model selection part into two parts. One is the best distribution model fitting for the data set, called automatic model selection module. The other part is to combine the learned index with the automatic model selection module to generate the best index for users.

3.1 Automatic Model Selection Module

Considering the diversity of data stored in the database, it is difficult to ensure that the exact location is always queried at a faster speed and high accuracy if the same model has been used. Therefore, in ALMSS, we propose to use multiple regression models with diversity and accuracy guarantees such as linear regression, polynomial regression, elastic regression, etc. At the same time, we use the random forest as the classifier, so as to ensure that ALMSS has fast and accurate characteristics in the diversity of data for the whole database. The specific measures are as follows. Firstly, for the received key range input, training is carried out in the existing multiple regression models. We obtain the evaluation data of the regression effect of each model, such as R2-score, Median absolute error, RMSE, etc. Then we establish the evaluation function and use random forest classifier for classification training. Finally we optimize the model performance. The model trained in this way, in the face of new data test, also has the ability to select the data characteristics of the excellent model regression analysis. At the same time, we note that due to the establishment of an excellent evaluation function, the Median absolute error of the trained model is very low, which also indicates that our model has excellent regression performance.

3.2 Learned Index with Automatic Model Selection Module

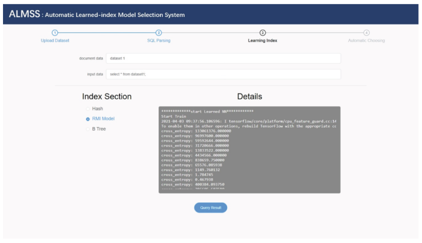
The existing leaf nodes of learning index are fixed single models, such as linear regression. This is not suitable for all datasets. At the same time, when the data distribution is complex, the model of leaf node using artificial selection is inefficient and it is not easy to choose. Therefore, on the basis of learning index, we change the leaf nodes in the learning index from the fixed model to our automatic model selection module. First, the RMI module divides the dataset into smaller ones. Then, on a small-scale dataset, we can use the machine learning method to automatically fit the most suitable model for this dataset as the leaf node model. It is worth noting that over-fitting is allowed in the process of training the model of leaf node. Overfitting means that the model we choose fully fits the dataset corresponding to the model. This is allowed in the database. We also use the same strategy as the original learning index. We set an error threshold for leaf nodes in advance. If the actual error is greater than the threshold we set, the leaf node will degenerate into a traditional index such as B-tree. If the actual error is within the threshold range we set, we use the automatically generated model as the leaf node. This ensures the accuracy of the index and improves the efficiency of the index.

4 Demonstration Scenarios

We plan to demonstrate our ALMSS system in four steps:

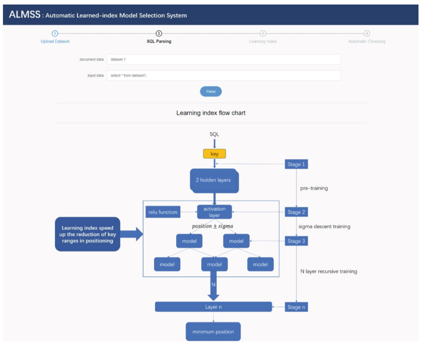
- Upload Dataset. As in Fig. 2(a), we provide an operation button and information display interface. Users can browse the existing dataset and select the data set they need according to the content of the dataset, data distribution and other information. At the same time, users can choose which index structure to use. The currently available index structures are B-Tree, hash, and learning index.

– Sql Parsing. As in Fig. 2(b), users enter SQL statements that need to query, and then our system will be regular parsed. We associate SQL statements with corresponding fields and calculate a large range of key. At the same time below the page, we give the diagram of the whole process.
– Learned Index. As Fig. 2(c), ALMSS provides three schemes for rapidly reducing the range of key. We use the recursive learning index technology (RMI Model), and compare it with the traditional B-tree and hash index. Users can view the time and other information of the visual index establishment process of different schemes provided by us.
– Automatic Selection. As Fig. 2(d) presents, ALMSS automatically selects the appropriate regression model for training in the key range. Users can see some relevant information such as the selected optimal model, model-related parameters, training and evaluation results, and the precise location of the final query (Fig. 2).

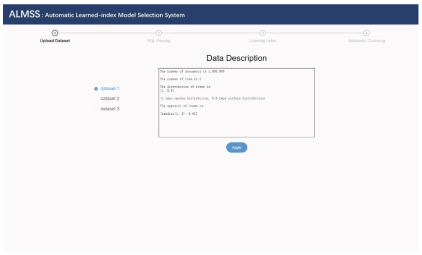


```
*************start Learned NN************
Start Train
2021-04-03 09:37:56.106596: I tensorflow/core/platform/cpu_feature_guard.cc:14
To enable them in other operations, rebuild TensorFlow with the appropriate cc
cross_entropy: 133061376.000000
cross_entropy: 96997600.000000
cross_entropy: 59592644.000000
cross_entropy: 31720666.000000
cross_entropy: 13833522.000000
cross_entropy: 4434566.000000
cross_entropy: 838659.750000
cross_entropy: 65576.085938
cross_entropy: 1149.760132
cross_entropy: 1.784745
cross_entropy: 0.467938
cross_entropy: 400384.093750
```

(a) Upload Dataset

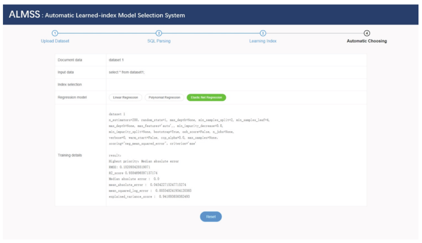


(b) Sql Parsing



```
The number of documents is 1,000,000

The number of item is 2

The distribution of items is:
[1, 0.0]

(1 reps random distribution, 0.0 reps uniform distribution)

The sparsity of items is:

[randint(1, 3), 0.01]
```

(c) Learned Index



Document data: dataset 1
Input data: select * from dataset1;
Index selection:
Regression model: Linear Regression | Polynomial Regression | [✓ Elastic Net Regression]
Training details:

```
dataset 1
n_estimators=200, random_state=1, max_depth=None, min_samples_split=2, min_samples_leaf=4,
max_depth=None, max_features='auto',, min_impurity_decrease=0.0,
min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None,
verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None,
scoring="neg_mean_squared_error", criterion='mse'

result:
Highest priority: Median absolute error
RMSE: 0.182093428519071
R2_score 0.9384696597137174
Median absolute error :  0.0
mean_absolute_error :  0.045422715247715274
mean_squared_log_error :  0.005548241934120363
explained_variance_score :  0.941693656362493
```

(d) Automatic Selection

Fig. 2. Demonstration scenarios

References

1. Kraska, T., Beutel, A., Chi, E.H., Dean, J., Polyzotis, N.: The case for learned index structures. In: Proceedings of the International Conference on Management of Data, 489–504 (2018)

2. Li, P., Hua, Y., Zuo, P., Jia, J.: A scalable learned index scheme in storage systems. arXiv: Databases (2019)
3. Ding, J., Minhas, U.F., Yu, J., Wang, C., Do, J., Li, Y.: ALEX: an updatable adaptive learned index. In: International Conference on Management of Data, pp. 969–984 (2020)
4. Ferragina, P., Vinciguerra, G.: The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds. Proc. VLDB Endow. **13**(8), 1162–1175 (2020)
5. Tang, C., Wang, Y., Hu, G., Dong, Z., Wang, Z., Wang, M.: XIndex: a scalable learned index for multicore data storage. In: ACM (2020)
6. Galakatos, A., Markovitch, M., Binnig, C., Fonseca, R., Kraska, T.: FITing-tree: a data-aware index structure. In: International Conference on Management of Data, pp. 1189–1206 (2019)