Knowledge and Information Systems (2021) 63:971–992
https://doi.org/10.1007/s10115-021-01547-7

REGULAR PAPER

Selectivity estimation with density-model-based multidimensional histogram

Meifan Zhang[1] · **Hongzhi Wang**[1,2]

Received: 4 July 2020 / Revised: 29 December 2020 / Accepted: 4 January 2021 /
Published online: 2 February 2021

Abstract
Histograms are widely used in selectivity estimation for one-dimensional data. Using the one-dimensional histograms to estimate the selectivity of the multidimensional queries will result in a high estimation error, unless the assumption of attribute independence is true. Constructing a multidimensional histogram also brings great challenges. The storage of a multidimensional histogram exponentially increases with the number of dimensions. In this paper, we propose a density-model-based multidimensional histogram. It uses a lightweight density model to predict the densities of a large number of regions instead of storing too many buckets. The experimental results indicate that our method can provide highly accurate selectivity estimations while occupying little space. In addition, the superiority of our method is more evident in high-dimensional data.

Keywords Selectivity estimation · Multidimensional histogram · Query processing

1 Introduction

Selectivity estimation is an important task in query optimization. The accuracy of selectivity estimation influences the efficiency of a query plan. Many synopses-based selectivity estimation approaches, such as sampling [24,34], histogram methods [33,36], wavelet methods [22] and kernel density estimation methods [11,13], have been proposed. Among these methods, histogram is the most popular method to solve this problem due to its compactness, efficiency and the variousness of applications [15].

Histograms partition the data points into buckets, and store the number of data points in each bucket along with the range of each bucket. The study of histogram starts from the one-dimensional histograms [15]. The one-dimensional query selectivity can be easily estimated based on the one-dimensional histograms. Constructing the one-dimensional histograms such as the equi-width histogram [20], equi-depth histogram [28], V-optimal histogram [16] and

✉ Hongzhi Wang
wangzh@hit.edu.cn

1 Department of Computer Science and Technology, Harbin Institute of Technology, Harbin, China

2 Peng Cheng Laboratory, Shenzhen, China

some variations [9,36] are well studied in previous researches, therefore, we will not discuss much about this problem here.

Multidimensional selectivity estimation is a problem very different from the one-dimensional version. Of course, we can also estimate the multidimensional selectivity estimation based on the one-dimensional histograms according to the assumption of attribute independence. However, this assumption is always violated in real-life data, especially in big data. As the number of dimensions increases, there may exist embedded associations among different attributes.

Constructing a multidimensional histogram is a good way to summarize the multidimensional data. However, constructing a multidimensional histogram brings more challenges than constructing a one-dimensional histogram. Usually, it is not easy to extend the methods of constructing a one-dimensional histogram to accommodate to the multidimensional data. First, high dimensions result in a high degree of freedom for bucketing. The bucketing schemes for multidimensional data are not only about choosing the bucket boundaries, but also related to the order of dimensions to deal with and the number of buckets in each dimension. Second, the storage of a multidimensional histogram is costly, since the space cost of a histogram exponentially increases with the number of dimensions. For example, if we equi-width bucket each dimension into ten parts, a 3D histogram will be composed of one thousand buckets. Furthermore, the information in each bucket also increases with the number of dimensions. In addition, there is a lot of space containing no data points in real-life multidimensional data, suggesting that using grid buckets such as the equi-width and equi-depth histograms will result in a great waste of storage. Third, the accuracy of estimating a multidimensional selectivity based on a histogram with limited number of buckets is not satisfactory [6]. The data points in the multidimensional space are more likely to be non-uniformly distributed, so it is difficult to separate the data points into uniform-density buckets. Consequently, the selectivity estimation based on the uniform-density assumption is not reliable.

The accuracy of selectivity estimation based on existing histograms is not satisfactory. In general, there are two ways to increase the accuracy of a histogram. The first one is to find an appropriate bucketing scheme which raises the uniformity of the data inside each bucket. But it is difficult to find such an optimal bucketing scheme for the multidimensional data. Even constructing a 2D V-optimal histogram is NP-hard [26]. The second one is to increase the number of buckets. As the number of buckets increases, the number of data points in each bucket reduces. Consequently, the estimation error brought by the uniform-density assumption reduces. The second way is more practical, but it is on the cost of increasing the histogram size at the same time.

We attempt to construct a compact and accurate histogram for multidimensional selectivity estimation. However, it is difficult to achieve these advantages at the same time. Usually, a histogram with more buckets and more information in each bucket leads to a more accurate estimation. Nevertheless, we do not want to raise the accuracy at the price of increasing the space cost. We consider to form a synopsis embedding the density information of a large number of buckets without occupying much space. Using machine learning models to replace some buckets seems to be a good way to reduce the space cost, due to their ability to model a lot of information while occupying little space.

The simplest idea of using machine learning models to handle the selectivity estimation is to train a regression model mapping each query in the workload to its selectivity. But tuning a good model for a high-dimensional data requires a large number of pre-computed queries. And executing those queries on the big data is a huge cost. In addition, if a new query does not follow the distribution of the pre-computed queries, the prediction of the new query based on the model is not reliable. In order to avoid collecting the costly pre-computed queries, we would like to learn a model from the data instead of the queries.

We propose a method called the density-model-based multidimensional histogram (DMMH) to meet these needs. We train a density model to predict the densities of some buckets instead of storing too many buckets. It saves a lot of space without reducing the accuracy. DMMH also stores some back-up buckets that cannot be predicted by the density model. DMMH estimates the selectivity of a query by summing up the densities of the regions predicted by the density model and the densities of the back-up buckets in the query range.

The idea of our DMMH starts from learning a model mapping each equi-width bucket to its density, since equi-width bucketing is the simplest and efficient bucketing scheme which can be accomplished with one pass of data. Besides, the equi-width buckets are easy to be encoded, because the shape of all the buckets is the same. Furthermore, a density model is able to restore the information of the equi-width buckets, since the bucket boundaries can be computed without accessing the data as long as the data domain is known. There is still a problem of a multidimensional equi-width histogram, that a large number of buckets are empty. Fitting the density model to those empty buckets is a huge waste, suggesting that, it requires a larger model or longer training time to tune the model well. In addition, the prediction error of the empty buckets will bring much error to the selectivity estimation result. In order to avoid increasing the difficulty of training the model, we only train the density model for a subset of the equi-width buckets. We find the dense regions where most of the data concentrates by clustering. We then train a density model predicting the density of each bucket in the dense regions. Finally, we store some back-up buckets not in the dense regions. In this way, we make use of both the machine learning model and some equi-width buckets to form a compact synopsis.

We make the following contributions in this paper.

- We proposed a density-model-based multidimensional histogram (DMMH). We use a data-driven machine learning model to replace a large number of equi-width buckets.
- We compare our DMMH with the sampling-based estimator and some typical multidimensional histograms including the equi-width histograms, equi-depth histograms, GenHist and STHoles. The experimental results demonstrate that our method outperforms these synopses. Our DMMH is more accurate while occupying less space. The superior of our method is more evident for high-dimensional queries.

The remaining parts of this paper are organized as follows. In Sect. 2, we survey the related works for this paper. In Sect. 3, we introduce the construction of DMMH and the selectivity estimation based on DMMH. In Sect. 4, some extensions are introduced. Section 5 shows the performance of our method. Finally, we conclude this study in Sect. 6.

2 Related work

Histograms are the most widely used synopses in selectivity estimation. They are good statistics to summarize data. In [29], the authors introduced several classes of histograms offering high accuracy for various estimation problems. Some widely used one-dimensional histograms including equi-width histograms, equi-depth histograms and V-optimal histograms are introduced in a survey of synopses [6].

Constructing a multidimensional histogram brings more challenges. As we discussed in the introduction, most of the one-dimensional histograms are not easy to accommodate to the multidimensional data except the equi-width histogram. However, the accuracy of a multidimensional equi-width histogram is unacceptable, unless the data distribution is uniform [6]. Most of the existing multidimensional histograms attempt to separate the data into buckets with close-to-uniform density, since the density of a region inside a bucket is estimated according to the uniform-density assumption. The multidimensional equi-depth histogram [25] arbitrarily chooses one dimension and separates the data along this dimension into equi-depth buckets. Then, each bucket is continuously split according to the remaining dimensions in the same way. Each dimension is split into $B^{1/d}$ buckets, where B is the number of buckets, and d is the number of dimensions. MHIST [30] starts with one bucket, and greedily chooses one dimension and one bucket to partition at one time. In each step, it uses the one-dimensional bucketing scheme, such as MaxDiff and V-optimal, to find the bucket in most need of partitioning. Min-Skew [2] supports selectivity estimation for spatial queries. It uses a greedy approach to partition the existing region into two in each step, while trying to minimize the spatial-skew of the grouping.

GenHist [10] allows overlapping buckets. It iteratively (1) partitions the data into equi-width buckets, (2) finds the dense buckets whose densities are higher than the average of their adjacent buckets, (3) stores each dense bucket and sets its density as the number of the elements exceeding the average density of its neighboring buckets and (4) removes the tuples in the dense buckets exceeding the average density of their adjacent buckets from the data. The construction process keeps storing the dense region and removing some data in the dense regions from the dataset, which makes the remaining data more uniformly distributed. Since the selectivity is estimated according to the uniform-density assumption, more uniformly distributed data leads to more accurate estimation results. But it requires several passes of data while the equi-width and equi-depth histograms require only one-pass of data. Consequently, the construction time of a GenHist is higher than the equi-depth histograms. In addition, since the GenHist allows overlapping buckets, it has to scan all the buckets in order to answer a query. Thus, its estimation efficiency is lower than the equi-width histograms.

Besides the static histograms built from the data, there are some self-tuning histograms using query results to dynamically build the histogram. The self-tuning histogram is first proposed in reference [1]. It uses query feedback to refine buckets. STHoles [4] allows buckets to have ‘holes’ rather than the overlapping of buckets. Different from other bucketing schemes, the number of tuples in each bucket in this histogram is computed according to the query feedback. It outperforms other data-driven histograms in most cases. But it does not work well for high-dimensional data, since its ability to capture the pattern based on the workload is diminished for multidimensional data. Because of its construction mechanism, a self-tuning histogram is sensitive to the queries order, suggesting that, changing the order of the learning queries has a significant impact on the histogram precision [18]. That is, a bad structure in the beginning will result in the local optima with high estimation errors. In addition, the STHoles traces the differences between the execution and estimated feedback at individual bucket level of the histogram, which increases the accuracy and also raises the overhead of execution time [5,17].

Nowadays, the machine learning methods are adopted to database. Some works use machine learning methods to solve the problem of selectivity estimation [7,12,27,35]. Most of the existing learning-based methods are query-driven methods [7,19,27], which learn models from the true selectivity of each query in the workload. The benefits are the high estimation efficiency and its ability to handle multidimensional range queries. However, collecting the training set is often expensive when the workload is unavailable or not sufficient, since the queries are required to be executed on big data. QuickSel [27] is a query-driven selectivity learning framework, which builds an internal model of the underlying data. It is reported to be more efficient than the query-driven histograms. Reference [19] builds a multiset convolution network (MSCN) on the sampling-based estimation to solve cardinality estimation. Some data-driven methods are proposed in recent years, most of them attempt to capture the joint probability distribution from a sample of data. They do not require the query workload, but their performance largely depends on the sample. Some recent works [12,35] proposed unsupervised learning methods which learn the joint probability distribution based on the autoregressive density estimator MADE. DeepDB [14] proposed a class of deep probabilistic models based on the relational sum product networks (RSPNs).

Usually, there is a trade-off between the accuracy and the space cost. The accuracy increases with the number of buckets and the information stored in each bucket. If we want to increase the accuracy without raising the space cost, we need more compact histograms.

3 DMMH: density-model-based multidimensional histogram

The structure of DMMH and some definitions will be introduced in Sect. 3.1. The construction of DMMH will be shown in Sect. 3.2. At last, we will introduce the selectivity estimation based on DMMH in Sect. 3.3.

3.1 Definitions and framework

As we introduced before, the density-model-based multidimensional histogram (DMMH) is composed of a density model, some dense regions and some buckets out of the dense regions. Each bucket in DMMH is in the form of a hyper-rectangle with multidimensional boundaries, and the count of data points in that bucket.

We first give some definitions before introducing the construction of DMMH.

We define a d-dim bucket and its density as follows.

Definition 1 d-dim bucket: a d-dimensional rectangle whose boundary is $[l_1, r_1, l_2, r_2, \ldots, l_d, r_d]$. The l_i and r_i mean the left and right boundaries of the ith dimension. We call the distance between l_i and r_i the length of the bucket in the ith dimension.

Definition 2 The density of a bucket: the number of data points inside the bucket.

The boundary and the density are the total information of a bucket in a multidimensional histogram. In order to avoid the huge cost of storing too many buckets, we would like to use a regression model to replace some buckets. The main tasks before getting the training data include choosing the bucketing scheme and encoding each bucket. We first consider the most common d-dim bucket, the regression model need to map its $2d$-dimensional boundary to its one-dimensional density. Thus, we need to compute the densities of a large number of buckets chosen randomly from the data domain to tune an accurate model. It is the same with the workload-based methods, if the densities of these buckets are known. However, if the workload is not available in advance, computing the densities on big data is costly. Modeling a histogram including the non-overlapping buckets in different shapes is not feasible. We take the equi-depth histogram for an example. An equi-depth histogram separates the data into buckets with the same density. Thus, the boundaries of the buckets cannot be computed without accessing the data, since the boundaries are related to the data. We cannot use a regression model to restore the information of the buckets in an equi-depth histogram. In addition, there is no need to predict the densities of the equi-depth buckets, since their densities are the same.

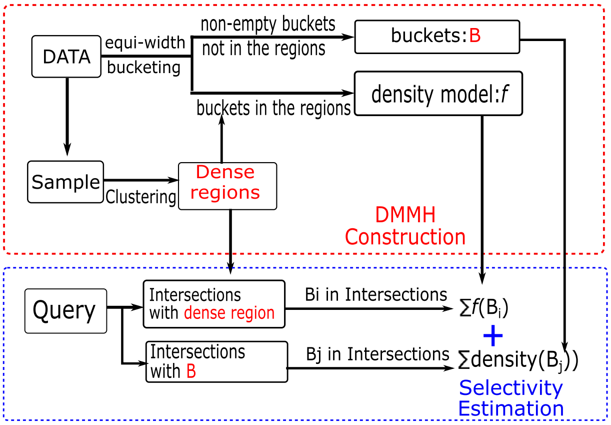


Fig. 1 Framework

We finally choose the simplest equi-width histogram, because the information of an equi-width bucket can be ideally restored by a regression model. The boundaries of the equi-width buckets can be computed directly as long as the data domain is available and the number of buckets in each dimension is known. And the densities of the buckets can be predicted by the regression model. Thus, we do not need to store the histogram along with the density model any more. In addition, the density model can be simplified by reducing the dimensions of the boundary information. Since the equi-width buckets are in the same shape, we can use the d left boundaries $[l_1, l_2, \ldots, l_d]$ of a bucket to represent the bucket. We call the set of these left boundaries the start point of the bucket, and define it as follows.

Definition 3 The start point of a d-dim bucket: the set of the left boundary in each dimension $[l_1, l_2, \ldots, l_d]$.

Thus, the density model can be trained by mapping the start point of each equi-width bucket to its density. The densities of the buckets can be computed with one pass of data.

After introducing the definitions about our new histogram DMMH, we will introduce the framework of our DMMH. The framework of our DMMH is shown in Fig. 1. The red dotted line fences the construction of the DMMH. We cluster a sample of the data to find the dense regions. We partition the data with an equi-width bucketing scheme. We count the densities of a subset of the equi-width buckets in the dense regions, and train a density model by mapping the start point of each bucket to its density. The other non-empty buckets out of the dense regions are stored in the DMMH. Thus, the DMMH is finally composed of a density model and a small number of multidimensional buckets. The blue dotted line fences the selectivity estimation based on the DMMH. The estimation is computed by summing up the density of each bucket in the query range. The details of constructing a DMMH and the selectivity estimation based on a DMMH will be introduced in Sects. 3.2 and 3.3, respectively.

3.2 The construction of a DMMH

We divide the construction of DMMH into three steps.

In the first step, we find the dense regions where most of the data concentrates. We get a sample from the data, and we cluster the sample with the Mini-Batch-Kmeans [32] method.

The Mini-Batch-Kmeans reduces the computation cost by orders of magnitude compared to the classic batch clustering algorithm, and the clustering result is comparable with other clustering methods. We calculate the boundary of each cluster, and regard each one as a dense region. Since a large number of regions are empty in the multidimensional data, tuning a density model fitting all the regions will increase both the model size and training difficulty. In addition, the prediction error of a large number of empty buckets will bring much estimation error. Therefore, training a model for the dense regions is a good choice to form a lightweight synopsis. In general, the dense regions do not cover all the data, since they are captured by the sample. The data outside the dense regions will be stored in some multidimensional buckets.

In the second step, we partition the data with an equi-width bucketing scheme, and the buckets in the dense regions will be used to train the density model. We adopt the equi-width bucketing scheme, because it is simple, efficient and can be well restored by a regression model. Given the data domain and the number of buckets in each dimension, the density of each bucket can be computed with only one pass of data. We train a density model mapping the start point of each bucket in the dense regions to its density. In the implementation, we use the index of the start point to represent the bucket.

In the third step, we store the non-empty buckets outside the dense regions. The density of these buckets can be computed along with the equi-width bucketing in the second step. Thus, the densities of any regions in the data domain can be estimated by either the density model or these back-up buckets. Including these non-empty buckets in the DMMH is cost-effective, since these buckets are only used to store the rare data outside the dense regions. That is, storing these buckets will not cost too much space. However, without these non-empty buckets for the outliers outside the dense regions, the dense regions will be extremely enlarged and contain much more empty regions resulting in a high prediction error.

Algorithm 1 DMMH-Construction

```
1: **Input: DATA** $D$**, Clusters** $|C|$**, EW-Buckets** $B$**, dimension** $d$
2: **Output: model** $f$**, back-up buckets** $UnitB$, DenseRegions $[R1, R2, ..., R|C|]$
3: Get the Sample $S \leftarrow D$
4: Clustering $S$ in to $|C|$ clusters with Mini-Batch-Kmeans
5: **for** each cluster $C_i$ **do**
6:     DenseRegions $R_i = Boundary(C_i)$
7: Training Data $Position$ =[ ], $Density$ =[ ]
8: Back-up Buckets $UnitB$ =[ ]
9: Partition $D$ into $B^d$ equi-width buckets.
10: $Left = [l_1, l_2, \dots, l_d]$
11: $Unit = [\frac{r_1-l_1}{B}, \frac{r_2-l_2}{B}, \dots, \frac{r_d-l_d}{B}]$
12: **for** each equi-width bucket $b_i$ inside DenseRegions **do**
13:     $StartIndex$ =[ ]
14:     **for** dimension $j$ in $d$ **do**
15:         $StartIndex += \frac{b_i[2\cdot j]-Left[j]}{Unit[j]}$
16:     $Position += StartIndex$
17:     $Density += density(b_i)$
18: **for** each non-empty bucket $b_i$ outside DenseRegions **do**
19:     $UnitB += b_i$
20: Train density model $f$ : $Position \rightarrow Density$
21: **Return:** $f$**,** $UnitB$**,** $[R_1, R_2, \dots, R_{|C|}]$
```

The pseudo-code in Algorithm 1 introduces the construction of DMMH. The $|C|$ means the number of clusters, B means the number of buckets in each dimension of the initial equi-width histogram. The first step starts from choosing a random sample from the data (Line 3). It then clusters the sample with the Mini-Batch-Kmeans method (Line 4). The boundary of each cluster fences a dense region (Line 5–6). The next step is to prepare the training set for the model. We would like to train a density model mapping the position of each equi-width bucket to its density. The algorithm initializes the training sets as empty sets (Line 7). As we mentioned before, we also need some back-up buckets outside the dense regions (Line 8). The data are then partitioned into B^d equi-width buckets (Line 9). We only care about the buckets inside the dense regions and the non-empty buckets outside the dense regions. We use the index of the start point of each equi-width bucket to denote the position of each bucket. The $Left$ is the set of the left boundary of the dataset in each dimension (Line 10), and the $Unit$ is the set of the unit length in each dimension (Line 11). The $Left$ and the $Unit$ are prepared to compute those indices. The algorithm computes the position and the density of each equi-width bucket b_i in the dense regions (Line 12–17). The boundary of b_i is a $2d$-dimensional value $[l_1, r_1, l_2, r_2, \ldots, l_d, r_d]$, and we use $b_i[2 \cdot j]$ to denote the left boundary of bucket b_i in dimension j (Line 15). We will use an example to show how to get the index of a bucket later. The non-empty equi-width buckets outside the dense regions are stored in the DMMH (Line 18–19). The density model is finally trained by mapping the $Position$ to the $Density$ (Line 20).

In this way, an equi-width histogram is transformed into a regression model f, some back-up buckets and some dense regions. We use the following example to show some details in the construction of DMMH.

Example 1 Figure 2 shows the process of constructing a DMMH for the 2D data. This figure simply illustrates the process of equi-width bucketing, data clustering, training the model and getting the back-up buckets. The red rectangles are the boundaries of three clusters, they fence the dense regions of the data. Each bucket in the grid is an equi-width bucket. The buckets inside the dense regions are used to train the density model. The non-empty buckets outside the dense regions (shown as the blue dotted rectangles) are also maintained in the DMMH.

In the implementation of this algorithm, the training set for the density model can be captured with one pass of data. We use the following example to show the location of a given data point and the index of a bucket.

Example 2 Figure 3 shows a 2D equi-width histogram of a dataset whose boundary is [1.2, 2.0, 0.1, 0.3]. The start point of the dataset is $Left = (1.2, 0.1)$, and the unit of each dimension is $Unit = [0.4, 0.1]$. The index of a bucket is the index of its start point. Suppose we want to find the index of the bucket where a data point $d(1.70, 0.25)$ locates, the index can be computed as follows.

$$StartIndex = \left(\left\lfloor \frac{d[0] - Left[0]}{Unit[0]} \right\rfloor, \left\lfloor \frac{d[1] - Left[1]}{Unit[1]} \right\rfloor \right) = (1, 1). \tag{1}$$

As shown in the example above, the location of each data point can be found in $O(1)$. Thus, the density of each bucket can be computed with one pass of data, no matter it is in the dense regions or not. Since we do not care about the empty buckets out of the dense regions, we use a HashMap to store the non-empty buckets outside the dense regions. Once a data point locates in a bucket outside the dense regions, we insert the bucket into the HashMap or increase its density in the HashMap. Thus, both the training set and the back-up buckets can be obtained in linear time.

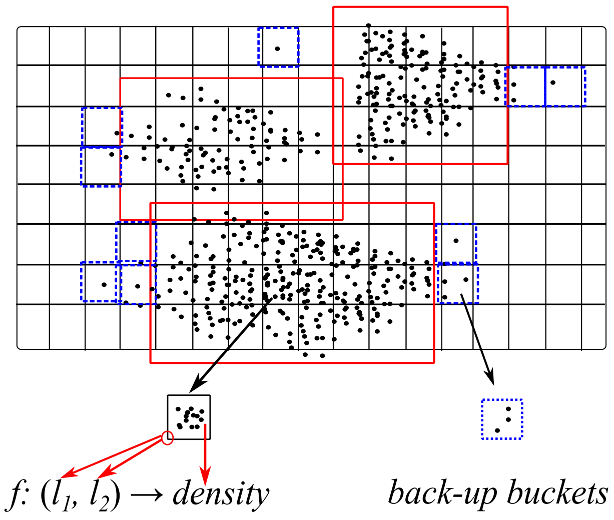
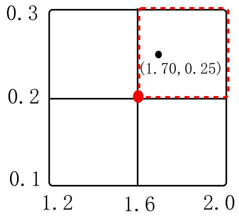


Fig. 2 The construction of DMMH

Fig. 3 The index of a bucket

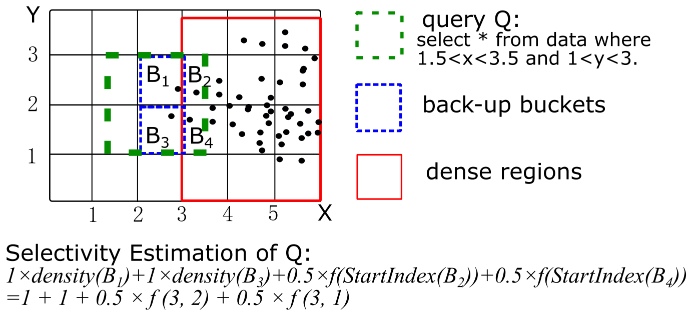


Using a density model instead of storing the buckets in the dense regions largely reduces the size of the synopsis. But we do not completely abandon the equi-width buckets, we also stored some back-up buckets outside the dense regions. The existence of some back-up buckets in the DMMH benefits both the model training process and the estimation accuracy. On the one hand, it is good for us to keep a lightweight model. If we enlarge the cluster boundaries to cover the back-up buckets, a large number of empty buckets will be involved into the dense regions. Thus, it may require a larger model fitting all these buckets. On the other hand, it avoids introducing the prediction error of too many empty buckets to the estimation. Since we compute the estimate by summing up the densities of all the equi-width buckets in the query range, the prediction errors of too many empty buckets influence the estimation accuracy.

Thus, the DMMH makes a combination of the equi-width buckets and a density model to form a highly accurate synopsis while occupying little space.

3.3 Selectivity estimation based on a DMMH

In this section, we introduce the selectivity estimation based on the DMMH. The algorithm estimates the selectivity of a query by summing up the densities of the equi-width buckets in the query range. The buckets inside the dense regions will be predicted by the density model, and true densities of the non-empty buckets outside the dense regions are stored in the back-up buckets.



Selectivity Estimation of Q:
$1\times density(B_1)+1\times density(B_3)+0.5\times f(StartIndex(B_2))+0.5\times f(StartIndex(B_4))$
$=1 + 1 + 0.5 \times f(3, 2) + 0.5 \times f(3, 1)$

Fig. 4 Selectivity estimation based on DMMH

Algorithm 2 describes the selectivity estimation based on the DMMH. The input includes the density model f, the back-up buckets $UnitB$, the dense regions $[R_1, R_2, \ldots, R_{|C|}]$ and a query Q. The estimation is initialized as zero (Line 3). Each bucket overlapping the query range contributes to the query result. We separate the query estimation into two steps. In the first step, the algorithm computes the density of the intersection between the query Q and the dense regions (Line 4–6). Since there exist some buckets partially overlap the query, we allocate a portion of its density to the query result according to the ratio of the intersection to the bucket range (Line 5). The density of each bucket in the dense regions is predicted by the density model according to the index of its start point (Line 6). In the second step, the algorithm computed the density of the intersection between the query Q and the non-empty buckets outside the dense regions (Line 7–9). The only difference of this step from the first step is that we can get the true densities of the non-empty buckets, since they are stored in the DMMH.

Algorithm 2 DMMH-SelectivityEstimation

```
1: Input: $f$, $UnitB$, DenseRegions$[R_1, R_2, \ldots, R_{|C|}]$, Query $Q$
2: Output: SelectivityEstimation $Est$
3: $Est = 0$
4: for each bucket $B_i$ overlaps the intersection of $Q$ and DenseRegions do
5:     $ratio = \frac{Q \cap B_i}{B_i}$
6:     $Est += ratio \cdot f(StartIndex(B_i))$
7: for each bucket $B_i$ overlaps the intersection of $Q$ and $UnitB$ do
8:     $ratio = \frac{Q \cap B_i}{B_i}$
9:     $Est += ratio \cdot density(B_i)$
10: Return: $Est$
```

We use the following example to show the process of selectivity estimation based on our DMMH.

Example 3 Figure 4 shows the query to be estimated and the DMMH of a 2D dataset. The DMMH is composed of the dense region fenced by the red rectangle, and the non-empty buckets outside the dense region (shown as the blue dotted rectangles). The query Q to be estimated is shown as the green rectangle. As shown in the figure, the query overlaps both the buckets B_2 and B_4 in the dense region and the non-empty buckets B_1 and B_3 outside the dense region. According to Algorithm 2, the densities of the non-empty buckets outside the dense regions are stored in DMMH. The density of the buckets in the dense regions are predicted by the density model f, the density model can predict the density of a bucket according to its $StartIndex$, which is introduced in Example 2. The selectivity of the query is estimated by summing the density of the intersection between the query and each bucket as shown in the figure. Each intersection is estimated according to the ratio of the overlap to the entire bucket based on the 'uniformly distributed' assumption. For example, the ratio of the overlap between the query and B_4 to the range of B_4 is 0.5, thus, the density of the intersection is estimated by $0.5 \times density(B_4)$. Since B_4 is in the dense region, its density is not stored in DMMH, but needs to be estimated by the density model f according to its $StartIndex$.

The space cost of the DMMH $Space(DMMH) = Space(model) + O(2 \cdot d \cdot B + B)$ includes two parts, i.e., the cost of the density model and the cost of the back-up buckets. Different kinds of models have different cost models, and the total space cost of the back-up buckets is $O(2 \cdot d \cdot B + B)$, where B is the number of buckets and d is the number of dimensions. Each bucket requires $2 \cdot d$ values for boundaries and one value for density.

The time cost of estimating the selectivity estimation based on our DMMH is $Time(DMMH) = B_{dense} \cdot Time(Predict) + B \cdot O(1)$, where B_{dense} is the number of equi-width buckets in the dense regions overlap the query, $Time(Predict)$ is the time cost of predicting the density of an equi-width bucket based on the density model, and B means the number of the back-up buckets. The selectivity estimation time includes (1) predicting the densities of the equi-width buckets in the intersection of the dense regions and the query and (2) searching the densities of the buckets outside the density region from the non-empty back-up buckets.

Even though the estimate is calculated by accumulation, the estimation process is efficient. First, the dense regions limit the buckets used to compute the estimation. Second, the densities of the buckets inside the dense regions could be predicted in parallel by the density model. Third, accessing the back-up buckets in a HashMap is efficient. Thus, both the densities of the buckets inside and outside the dense regions can be computed efficiently.

4 Extensions

Our DMMH models the densities of a large number of equi-width buckets. The estimation accuracy largely depends on the denseness of the initial equi-width histogram and the prediction accuracy of the density model. More buckets in the training set lead to a more accurate model, but it also results in increasing the difficulty of both collecting the training set and tuning the model. In this section, we propose some methods to reduce the training set and increase the accuracy when the model is not well tuned.

4.1 Cut down the training set

In our DMMH, we use the equi-width buckets inside the dense regions to train the model. However, since our dense regions are in the form of multidimensional rectangles, they are still possible to include many empty buckets. We would like to exclude some of the empty buckets not adjacent to non-empty buckets from the training set.

We use an example to illustrate this strategy. Suppose the girds in Fig. 5 are the buckets inside a cluster boundary of a 2D dataset. The regions labeled by the red filled circles are non-empty regions, the others are empty regions. The training set includes three kinds of regions, the non-empty ones (labeled by the red circles), the empty ones (labeled by the blue

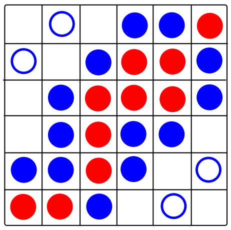


Fig. 5 Cut down the training set

filled circles) adjacent to the non-empty ones and some other empty regions randomly chosen from the remaining regions (labeled by the blue hollow circle). In this example, the last kind of regions are randomly chosen with a probability of $p = 1/3$. Even though this method only reduces a small number of records in the training set in this example, it will result in a significant reduction of the training set when dealing with a large high-dimensional data.

4.2 Add a filter to DMMH

The estimation accuracy based on the DMMH largely depends on the accuracy of the density model. Usually, a model in a small size is sufficient to provide satisfactory predictions for a large number of buckets. But we also consider the situation that the model is not well tuned to accurately predict the densities of all the buckets. We add a filter to the DMMH to store the densities of the buckets whose predictions are far from their true densities.

After training the model, we test the prediction error of the buckets in the dense regions. The buckets with high prediction errors will be inserted into the filter. The estimation process should also be modified. The intersection of a query and a bucket in the filter should be computed based on its true density stored in the filter instead of the prediction.

Adding a filter to the DMMH costs a little more construction time, but it will increase the accuracy of the DMMH.

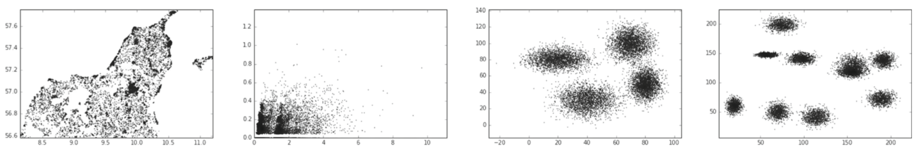
5 Experiments

In Sect. 5.1, we introduce the settings including the datasets, the accuracy metric and the implementation of the density model. In Sect. 5.2, we compare the performance of our methods with some previous methods. In Sect. 5.3, we make a conclusion of the experimental results.

5.1 Experimental settings

DataSets: We use three real datasets and two synthetic datasets to evaluate the performance. The synthetic datasets include GAUSS2D and GAUSS3D datasets. Both of these two datasets contain 10M tuples. The GAUSS2D and GAUSS3D are the combinations of four and ten different Gaussian distributions with different random means and standard deviations, respectively. The real datasets include the ROAD2D, ROAD3D and POWER7D datasets. The ROAD3D dataset[1] was constructed by adding elevation information to a 2D road network in North Jutland, Denmark. The dataset contains 434,874 tuples. The POWER7D dataset is

[1] http://archive.ics.uci.edu/ml/datasets/3D+Road+Network+(North+Jutland,+Denmark).



(a) ROAD3D **(b)** POWER7D **(c)** GAUSS2D **(d)** GAUSS3D

Fig. 6 The distribution of the datasets in the first two dimensions

a subset of a 9-dimensional dataset 'Individual household electric power consumption Data Set.'[2] This dataset contains 2,075,259 tuples and 9 attributes. The first two attributes of the 9D dataset are 'date' and 'time,' and we use the remaining seven numerical attributes to form the POWER7D dataset for our experiments.

Figure 6 shows the distribution of 10,000 samples from each dataset in the first two dimensions. The ROAD2D is the same with the first two dimensions of ROAD3D. Figure 6b shows the distribution of the first two numerical attributes 'global_active_power' and 'global_reactive_power' of the POWER7D dataset, and these two attributes mean 'the household global minute-averaged active power' and 'the household global minute-averaged reactive power,' respectively. Most of the minute-averaged power concentrate in a small region, therefore, the distribution shown in Fig. 6b is skew.

Metric: We use the Q-Error [23] to measure the accuracy of the selectivity estimation result due to its symmetry. It is widely used in selectivity estimation researches [7,12,24], and it relates to the query plan quality. In the following equation, $\hat{s}$ is a selectivity estimate, and s is the true selectivity.

$$\text{Q-Error} = \max\left\{\frac{s}{\hat{s}}, \frac{\hat{s}}{s}\right\}. \tag{2}$$

We choose queries with nonzero selectivity in the experiment and set the minimum estimation to be one, thus, none of the estimation and the true selectivity can be zero.

Query: The queries for the ROAD2D, ROAD3D, GAUSS2D and GAUSS3D were generated by randomly choosing the query boundaries from the data domain. We chose the queries with nonzero selectivity in the following experiments.

The queries for the POWER7D were generated in a different way. Since the data distribution is highly skew in high-dimensional data, the randomly chosen queries rarely have nonzero results. In order to avoid most of the generated query results to be zero, we limited the range to generate the query boundaries in each dimension for the POWER7D dataset. The left boundary of the query range in each dimension was randomly chosen from the first quarter of the entire value range of the corresponding attribute. Similarly, each right boundary was randomly chosen from the last quarter of the corresponding attribute range.

The Implementation of the Density Model: All the experiments were conducted in Python 3.5. We used the RandomForestRegressor in the scikit-learn package to model the relation between each bucket and its density. A random forest is a meta-estimator that fits a number of decision tree classifiers on various subsamples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [3]. We simply adopted the RandomForestRegressor since it is widely used and its parameters are easy to tune. We varied its parameters $n\_estimators$ and max_depth to form different density models.

[2] http://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption.

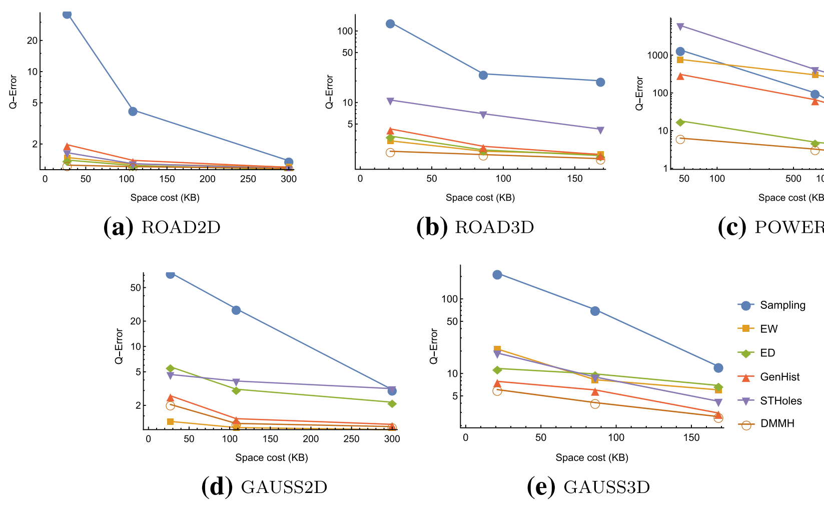


Fig. 7 The impact of synopses size on the accuracy

5.2 Experimental result

In this section, we compare the performance of our method with the sampling-based estimation (Sampling) and some typical multidimensional histograms including the equi-width histogram (EW), equi-depth histogram (ED), GenHist and STHoles. The sampling-based estimator is the baseline method for big data analysis. The EW and ED histograms are the most widely used histograms, GenHist is the most typical histogram allowing overlapping buckets, and STHoles is the state-of-the-art query-driven histogram. Some recent query-driven machine learning models for selectivity estimation, such as reference [7], are not included in the experiments. Because the query-driven models require a large number of pre-computed queries for training, and collecting the training data is expensive for big data.

In Sect. 5.2.1, we test the influence factors of the accuracy, including the synopsis size, the query selectivity, the denseness of the initial equi-width bucket and the number of clusters. In Sect. 5.2.2, we test both the construction efficiency and the estimation efficiency of different synopses. We test the impact of adding a filter to the DMMH in Sect. 5.2.3, and we report the performance of the strategy that cutting down the training set in Sect. 5.2.4.

5.2.1 Accuracy

EXP1: The impact of synopsis size on the accuracy.

Figure 7 shows the accuracy of these six synopses on the five datasets. We report the average of the Q-Error of 100 queries for each dataset. The lower the Q-Error, the higher the accuracy of selectivity estimation. We varied the parameters $n\_estimators$ from 10 to 100, and max_depth from 5 to 8 in the RandomForestRegressor to form density models with different sizes. The synopsis size of the EW, ED, GenHist and STHoles is determined by the number of buckets in these histograms. The synopsis size of the sampling-based estimator is the size of the sample.

We test the accuracy of these synopses in different sizes. It is clear that the accuracy increases with the synopsis size, since more samples or more buckets lead to more accurate estimation. Our DMMH outperforms the other methods on accuracy with the same space cost in most of the cases. We can also learn from the figure that the number of dimensions influences the accuracy of these synopses. The superior of our DMMH is more evident in handling the multidimensional data.

The EW, ED, GenHist and DMMH perform better than Sampling and STHoles for the 2D and 3D datasets in most cases. Sampling-based estimation is not reliable when the sample size is not sufficient. The histograms are more accurate for the low-dimensional data since they can provide some summarized statistical information. But the sampling-based estimation is more reliable for high-dimensional data when the sample size is large and enough. As shown in Fig. 7c, the sampling-based estimation is finally comparable to the DMMH when enough samples are provided. The reason is that the samples represent the distribution of the entire dataset, while the buckets only store the density of each bucket without the distribution inside the bucket. The accuracy of the STHoles largely depends on the workload, whose value is diminished for high-dimensional data. If the given query does not intersect with the workload, the STHoles has no help for the query. Most of the other synopses do not perform well for POWER7D dataset, because limited number of buckets cannot provide enough information for the high-dimensional selectivity estimation. However, the accuracy of our DMMH does not suffer much from the high dimensions, since we use a small model predicting the densities of a large number of buckets. Therefore, our DMMH performs better than the other synopses while occupying the same space.

All the synopses perform well for the ROAD2D dataset except the sampling-based method, because the distribution of the ROAD2D is more uniform compared with the other datasets as shown in Fig. 6. For the ROAD3D, GAUSS3D and POWER7D datasets with more skew distributions, the superior of our DMMH is more evident.

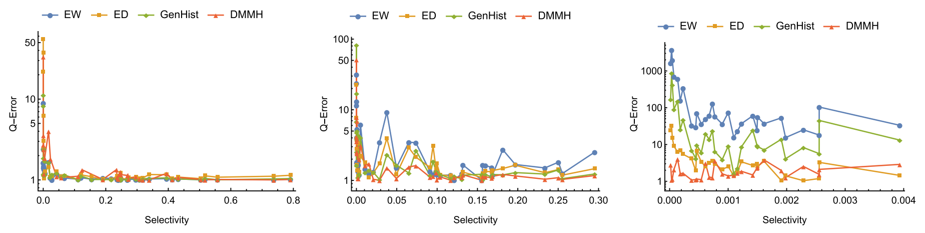
EXP2: The impact of selectivity on the accuracy.

Figure 8 shows the impact of the query selectivity on the accuracy of different synopses including the EW, ED, GenHist and our DMMH. The synopsis sizes for the 2D, 3D and 7D datasets are 27KB, 21KB and 787KB. The selectivities of the queries for the 7D dataset are very small due to our generation method introduced in Sect. 5.1. The sampling-based estimation and the STHoles are not involved in this experiment, since these synopses in a small size do not perform well.

We can learn from the figure that the accuracies of all the synopses increase with the selectivity. As the number of dimensions increases, the Q-Error of the EW, ED and GenHist have growth in different levels, but our DMMH does not grow as much. The difference of these synopses is more evident for 7D and 3D dataset compared with that for the 2D dataset. Our DMMH learns the information of a dense equi-width histogram with only a small model. Thus, it performs well for the low-selectivity queries. The error of a histogram-based estimation is caused by the assumption of uniformity. A query with a high selectivity is more likely to cover more complete buckets, thus, the region where the query partially intersects with a bucket occupies only a small part of the result. The performances of these synopses have little difference for the 2D dataset, but our DMMH is superior in the 7D selectivity estimation.

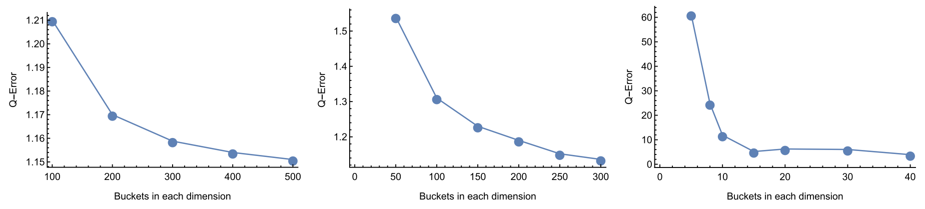
EXP3: The impact of bucketing denseness on the accuracy of DMMH.

Our DMMH uses an equi-width bucketing scheme to prepare the training data for the density model. The denseness of the initial equi-width bucketing scheme also influences the accuracy of the DMMH. In this experiment, we test the accuracy of DMMH with different bucketing densenesses. The experiment is conducted on the ROAD2D, ROAD3D and POWER7D datasets. The size of the DMMH for each dataset is 33KB. We varied the number of buckets in each dimension to form different bucketing densenesses. The result is shown in



(a) GAUSS2D **(b)** GAUSS3D **(c)** POWER7D

Fig. 8 The impact of selectivity on the accuracy



(a) ROAD2D **(b)** ROAD3D **(c)** POWER7D

Fig. 9 The impact of bucketing on accuracy

Fig. 9. We can learn from the figure that, the accuracy increases with the number of buckets in each dimension. The curve tends to be stable as the number of buckets increases to a large number in Fig. 9c. The reason is that the number of buckets in the dense regions increases with the number of buckets in the original equi-width histogram. Thus, the size of the training set increases accordingly. But such a small model is not possible to fit too many buckets well, its prediction error may increase with the number of buckets. Therefore, the rate of the Q-Error reduction slows down as the number of buckets increases to a large number.

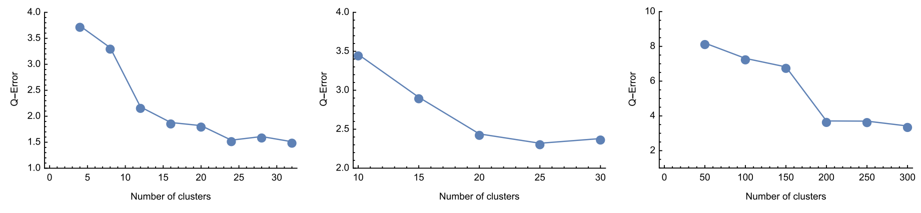
EXP4: The impact of clusters on the accuracy of DMMH.

We cluster a sample of data to find the dense regions for the DMMH. In this experiment, we test the impact of the number of clusters on the DMMH accuracy. The experiment was conducted on the ROAD2D, ROAD3D and POWER7D datasets. The number of buckets in each dimension of the initial equi-width histograms for these three datasets is 100, 50 and 20. Figure 10 reports the average Q-Error of 100 queries for each dataset. It shows that the Q-Error decreases with the number of clusters. As the number of clusters increases, the size of each cluster decreases. Since we use a hyper-rectangle to form the boundary of a cluster, a smaller cluster is more possible to include less empty equi-width buckets. Our DMMH estimates the densities of the initial equi-width buckets in the dense regions based on the density model predictions, therefore, the less empty buckets in the dense regions, the less prediction errors for these buckets.

5.2.2 Efficiency

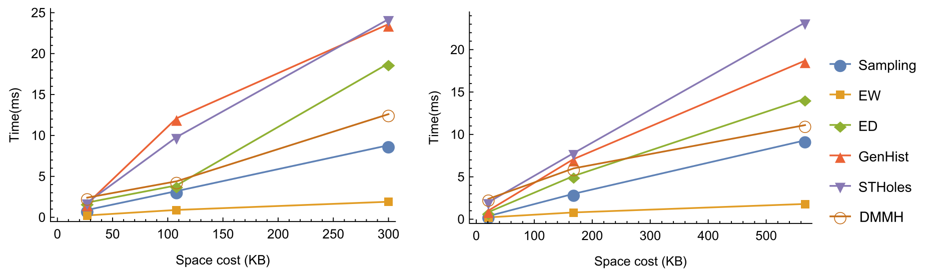
EXP1: The estimation efficiency.

We compare the efficiency of the different synopses, the result is shown in Fig. 11. We test the estimation time of 100 queries with these synopses, and report the average estimation time. The experiment is conducted on the GAUSS2D and GAUSS3D datasets. We can learn from the figure that the efficiency of our DMMH is comparable with other synopses. The



(a) ROAD2D **(b)** ROAD3D **(c)** POWER7D

Fig. 10 The impact of clusters on accuracy



(a) 2D **(b)** 3D

Fig. 11 Estimation efficiency

sampling-based estimation time is linear with the sample size. The EW is the most efficient synopsis in the figure, because it can directly find each bucket intersecting with a query according to the index. The ED and GenHist need to check the intersection between each bucket and the query, thus they spend more time than the EW. The STHoles spends even more time due to its tree structure. It is costly to deal with the intersection between a query and a bucket with a child. The efficiency of our DMMH is only a little worse than the sampling-based estimation.

EXP2: The construction efficiency.

In this experiment, we test the impact of the data size on the construction efficiency of different synopses. The STHoles is not involved in this experiment since it is a query-driven method. We compare the construction time of different synopses in the same size. Figure 12 reports the results. We can learn from this figure that the construction time increases with the data size. The GenHist spends most construction time, since it requires accessing the data several times, while the other methods only access the data one time. The EW spends more construction time than the ED due to the construction of the indices, which benefits the estimation efficiency. Our DMMH spends a little more time compared with the EW, due to the model training and clustering. Computing the bucket densities of the initial dense equi-width histogram does not cost much, since an equi-width histogram can be computed in one pass of data. A little longer construction time is acceptable, since the synopsis construction is done off-line, and the training and clustering do benefit the accuracy without reducing much estimation efficiency. As the researchers keep making efforts to provide new ML and clustering accelerators [8,21,31], the construction time of our DMMH will be further reduced in the future.

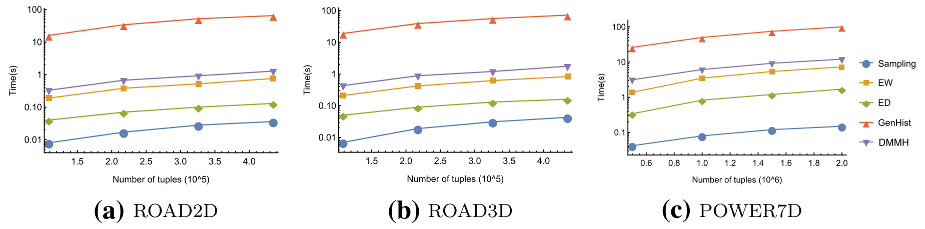


Fig. 12 Construction efficiency

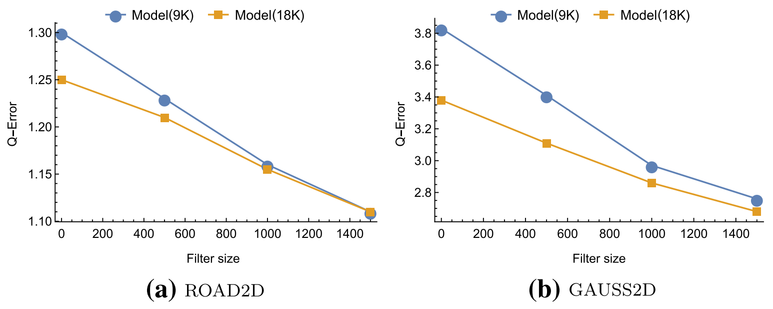


Fig. 13 DMMH with a filter

5.2.3 DMMH with a filter

In this experiment, we test the impact of adding a filter to a DMMH. We test the influence of the filter size on the accuracy of DMMH with density models in different sizes. This experiment is conducted on the real ROAD2D dataset and the synthetic GAUSS2D dataset. The number of buckets in each dimension of the initial equi-width histogram is 100. We use the average Q-Error of 100 queries to represent the accuracy of a DMMH. The result is shown in Fig. 13. The filter size in this figure means the number of buckets in the filter. We can learn from the figure that the Q-Error decreases with the filter size. The impact of the filter on the DMMH with a 9KB model is more evident than that on a DMMH with an 18KB model, because a larger model is more accurate and leaves little room for improvement.

5.2.4 Cut down the training set

In this experiment, we test the influence of cutting down the training set on the performance of DMMH as we mentioned in Sect. 4.1. We conduct a DMMH in the size of 8KB for the GAUSS2D dataset. We limit the synopsis in a small size to evidently show the change of the error. The sampling probability p of inserting the empty buckets into the training set varied from 0 to 1. Figure 14a, b shows the impact of p on the accuracy and the size of the training set, respectively. The trend of the Q-Error gradually slows down as p increases, while the size of the training set is in linear with p. Thus, the strategy can appropriately reduce the size of the training set without reducing much accuracy.

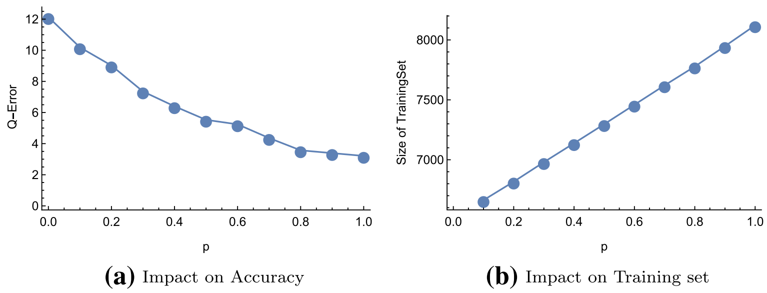


Fig. 14 The impact of p

5.3 Experimental summary

- Our DMMH can provide a more accurate selectivity estimation compared with some typical synopses including the sampling-based estimator, equi-width histogram, equi-depth histogram, GenHist and STHoles while occupying the same space.
- The efficiency of our DMMH is comparable to the other synopses.
- The accuracy superiority of our DMMH is more evident for the multidimensional query with a low selectivity.

6 Conclusions

In this work, we propose a density-model-based multidimensional histogram (DMMH) for selectivity estimation. This method makes use of machine learning to construct a density model for multidimensional data. It reduces the space cost without reducing the estimation accuracy. The experimental results demonstrate that our DMMH can provide more accurate selectivity estimation results with less space cost. In addition, our method is superior in handling multidimensional data. Our DMMH is a data-driven synopsis, and we will consider using it in conjunction with the query-driven methods in the future.

Acknowledgements This paper was supported by NSFC Grant U1866602 and CCF-Huawei Database System Innovation Research Plan CCF-HuaweiDBIR2020007B.

References

1. Aboulnaga A, Chaudhuri S (1999) Self-tuning histograms: building histograms without looking at data. In: SIGMOD 1999, proceedings ACM SIGMOD international conference on management of data, 1–3 June 1999, Philadelphia, Pennsylvania, USA, pp 181–192
2. Acharya S, Poosala V, Ramaswamy S (1999) Selectivity estimation in spatial databases. In: SIGMOD 1999, proceedings ACM SIGMOD international conference on management of data, 1–3 June 1999, Philadelphia, Pennsylvania, USA, pp 13–24
3. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
4. Bruno N, Chaudhuri S, Gravano L (2001) Stholes: a multidimensional workload-aware histogram. In: Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, 21–24 May 2001, pp 211–222
5. Chaudhuri S, Narasayya VR (2007) Self-tuning database systems: a decade of progress. In: Koch C, Gehrke J, Garofalakis MN, Srivastava D, Aberer K, Deshpande A, Florescu D, Chan CY, Ganti V, Kanne

C-C, Klas W, Neuhold EJ (eds) Proceedings of the 33rd international conference on very large data bases, University of Vienna, Austria, 23–27 Sept 2007. ACM, pp 3–14

6. Cormode G, Garofalakis MN, Haas PJ, Jermaine C (2012) Synopses for massive data: samples, histograms, wavelets, sketches. Found Trends Databases 4(1–3):1–294
7. Dutt A, Wang C, Nazi A, Kandula S, Narasayya VR, Chaudhuri S (2019) Selectivity estimation for range predicates using lightweight models. Proc VLDB Endow 12(9):1044–1057
8. Gao B, Liu N, Wang X, Lan M, Zhao Z, Dellandréa E, Chen L (2018) A method to accelerate k-means and GMM computation with GPU and multi-core CPU. In: Fourth IEEE international conference on multimedia big data, BigMM 2018, Xi'an, China, 13–16 Sept 2018. IEEE, pp 1–5
9. Guha S, Koudas N, Shim K (2006) Approximation and streaming algorithms for histogram construction problems. ACM Trans Database Syst 31(1):396–438
10. Gunopulos D, Kollios G, Tsotras VJ, Domeniconi C (2000) Approximating multi-dimensional aggregate range queries over real attributes. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data, 16–18 May 2000, Dallas, Texas, USA, pp 463–474
11. Gunopulos D, Kollios G, Tsotras VJ, Domeniconi C (2005) Selectivity estimators for multidimensional range queries over real attributes. VLDB J 14(2):137–154
12. Hasan S, Thirumuruganathan S, Augustine J, Koudas N, Das G (2020) Deep learning models for selectivity estimation of multi-attribute queries. In: Maier D, Pottinger R, Doan A, Tan W-C, Alawini A, Ngo HQ (eds) Proceedings of the 2020 international conference on management of data, SIGMOD conference 2020, online conference [Portland, OR, USA], 14–19 June 2020. ACM, pp 1035–1050
13. Heimel M, Kiefer M, Markl V (2015) Self-tuning, GPU-accelerated kernel density models for multidimensional selectivity estimation. In: Proceedings of the 2015 ACM SIGMOD international conference on management of data, Melbourne, Victoria, Australia, May 31–June 4, 2015, pp 1477–1492
14. Hilprecht B, Schmidt A, Kulessa M, Molina A, Kersting K, Binnig C (2020) Deepdb: Learn from data, not from queries!. Proc VLDB Endow 13(7):992–1005
15. Ioannidis YE (2003) The history of histograms (abridged). In: VLDB 2003, proceedings of 29th international conference on very large data bases, 9–12 Sept 2003, Berlin, Germany, pp 19–30
16. Ioannidis YE, Poosala V (1995) Balancing histogram optimality and practicality for query result size estimation. In: Proceedings of the 1995 ACM SIGMOD international conference on management of data, San Jose, California, 22–25 May 1995, pp 233–244
17. Kaushik R, Suciu D (2009) Consistent histograms in the presence of distinct value counts. Proc VLDB Endow 2(1):850–861
18. Khachatryan A, Müller E, Böhm K, Stier C (2016) Improving accuracy and robustness of self-tuning histograms by subspace clustering. In: 32nd IEEE international conference on data engineering, ICDE 2016, Helsinki, Finland, 16–20 May 2016, pp 1544–1545
19. Kipf A, Kipf T, Radke B, Leis V, Boncz PA, Kemper A (2019) Learned cardinalities: estimating correlated joins with deep learning. In: CIDR 2019, 9th Biennial conference on innovative data systems research, Asilomar, CA, USA, 13–16 Jan 2019, online proceedings. www.cidrdb.org
20. Kooi R (September 1980) The optimization of queries in relational databases. Ph.D. thesis, Case Western Reserve University
21. Low JS, Ghafoori Z, Bezdek JC, Leckie C (2019) Seeding on samples for accelerating k-means clustering. In: Proceedings of the 3rd international conference on big data and internet of things, BDIOT 2019, La Trobe University, Melbourne, VIC, Australia, 22–24 Aug 2019. ACM, pp 41–45
22. Matias Y, Vitter JS, Wang M (2000) Dynamic maintenance of wavelet-based histograms. In: VLDB 2000, proceedings of 26th international conference on very large data bases, 10–14 Sept 2000, Cairo, Egypt, pp 101–110
23. Moerkotte G, Neumann T, Steidl G (2009) Preventing bad plans by bounding the impact of cardinality estimation errors. Proc VLDB Endow 2(1):982–993
24. Müller M, Moerkotte G, Kolb O (2018) Improved selectivity estimation by combining knowledge from sampling and synopses. PVLDB 11(9):1016–1028
25. Muralikrishna M, DeWitt DJ (1988) Equi-depth histograms for estimating selectivity factors for multidimensional queries. In: Proceedings of the 1988 ACM SIGMOD international conference on management of data, Chicago, Illinois, USA, 1–3 June 1988, pp 28–36
26. Muthukrishnan S, Poosala V, Suel T (1999) On rectangular partitionings in two dimensions: algorithms, complexity, and applications. In: Database theory—ICDT '99, 7th international conference, Jerusalem, Israel, 10–12 Jan 1999, Proceedings, pp 236–256
27. Park Y, Zhong S, Mozafari B (2020) Quicksel: quick selectivity learning with mixture models. In: Maier D, Pottinger R, Doan A, Tan W-C, Alawini A, Ngo HQ (eds) Proceedings of the 2020 international conference on management of data, SIGMOD Conference 2020, online conference [Portland, OR, USA], 14–19 June 2020. ACM, pp 1017–1033

28. Piatetsky-Shapiro G, Connell C (1984) Accurate estimation of the number of tuples satisfying a condition. In: SIGMOD'84, proceedings of annual meeting, Boston, Massachusetts, USA, 18–21 June 1984, pp 256–276
29. Poosala V, Ioannidis YE (1996) Estimation of query-result distribution and its application in parallel-join load balancing. In: VLDB'96, proceedings of 22th international conference on very large data bases, 3–6 Sept 1996, Mumbai (Bombay), India, pp 448–459
30. Poosala V, Ioannidis YE (1997) Selectivity estimation without the attribute value independence assumption. In: VLDB'97, proceedings of 23rd international conference on very large data bases, 25–29 Aug 1997, Athens, Greece, pp 486–495
31. Reuther A, Michaleas P, Jones M, Gadepally V, Samsi S, Kepner J (2019) Survey and benchmarking of machine learning accelerators. In: 2019 IEEE high performance extreme computing conference, HPEC 2019, Waltham, MA, USA, 24–26 Sept 2019. IEEE, pp 1–9
32. Sculley D (2010) Web-scale k-means clustering. In: Proceedings of the 19th international conference on world wide web, WWW 2010, Raleigh, North Carolina, USA, 26–30 April 2010, pp 1177–1178
33. Shekelyan M, Dignös A, Gamper J (2017) Digithist: a histogram-based data summary with tight error bounds. PVLDB 10(11):1514–1525
34. Wu Y-L, Agrawal D, El Abbadi A (2002) Query estimation by adaptive sampling. In: Proceedings of the 18th international conference on data engineering, San Jose, CA, USA, February 26–March 1, 2002, pp 639–648
35. Yang Z, Liang E, Kamsetty A, Chenggang W, Duan Y, Chen P, Abbeel P, Hellerstein JM, Krishnan S, Stoica I (2019) Deep unsupervised cardinality estimation. Proc VLDB Endow 13(3):279–292
36. Yildiz B, Büyüktanir T, Emekçi F (2016) Equi-depth histogram construction for big data with quality guarantees. CoRR, arXiv:1606.05633

Meifan Zhang received the B.S. degree in Computer Science from Harbin Institute of Technology. She is currently a Ph.D. student at the School of Computer Science and Technology, Harbin Institute of Technology, China. Her research interests include big data analytics and data quality.

Hongzhi Wang is a professor and doctoral supervisor at the School of Computer Science and Technology, Harbin Institute of Technology, China. His research interests include big data management and analysis, data quality, graph data management, web data management. He has published more than 150 papers. He is the Primary Investigator of more than 10 projects including three NSFC projects, and co-PI of 973, 863 and NSFC key projects. He was awarded Microsoft fellowship, China Excellent Database Engineer and IBM Ph.D. fellowship. His Ph.D. thesis was selected as CCF outstanding Ph.D. thesis.