

## Selectivity Estimation of Range Queries over Data Streams Using Cosine Transform

Feng Yan, Wen-Chi Hou, Zhewei Jiang, Yong Huang, Dunren Che

Dept. of Compute Science Southern Illinois University, Carbondale, IL 62901 USA

{hou, zjiang, dche }@cs.siu.edu

**Abstract.** In this paper, we propose to approximate the cumulative distribution functions of data using the cosine transform. We develop a framework for estimating the selectivity of range queries over data streams using the cosine series. We show that the derivation of the cosine estimators is simple and efficient. The estimators can also be also updated easily in the presence of insertions and deletions. These features make the cosine transform most suitable for dynamic data stream environments. We conduct experiments to compare its performance with a well-known technique – the wavelet transform. The results show that cosine transform is superior to the wavelets in estimation accuracy and estimation and update speeds, especially in multi-dimensional cases.

**Keywords:** range query processing, selectivity estimation, cosine transform, stream data querying.

### 1 Introduction

In a data stream environment, data arrive at great speed and in large volume. To observe, monitor, or summarize continuous flow of data, queries are usually posted periodically. It is typically referred to as a continuous query because it is issued once and then logically run continuously over the data stream [2], unlike the traditional one-time query that is executed only once.

The processing of IP traffic data in the network [3, 19] is a classical data stream example. Routers forward and process IP packets at great speed for a variety of monitoring tasks, such as keeping track of statistics and detecting network attacks. Aggregation queries are very useful in

these scenarios in providing statistical summaries of the traffic carried by a link, identifying normal activities vs. activities under denial of service attack, etc.

There are two main constraints in continuous query processing: the amount of memory needed and the per-item processing time. It has been proved that without knowing the size of data streams, it is not possible to place a limit on the memory requirements for most queries unless the domains of the attributes involved in the query are restricted. For multiple aggregations, all tuples streamed by must be retained if exact results of the queries are demanded. Unfortunately, this may not be feasible because of the unboundness of data streams. Therefore, much research has focused on approximation or estimation of continuous queries, instead of exact query results.

Approximate aggregate query processing has various applications in conventional database research, such as selectivity estimation, query optimization, OLAP, decision support, etc. It has also found its place in data stream processing, for example, the trend analysis, fraud detection, quality/performance monitoring, etc.

Estimation of aggregate queries (or approximate aggregate query processing) has been an important research topic in conventional database research. Various techniques, such as sampling [1, 9, 13, 14, 21, 30], histogram [11, 15, 16, 17], wavelet [5, 6, 23, 29], discrete cosine transform [20] etc., have been proposed and some of them have been implemented in commercial database systems. In a continuous query-processing environment, approximate results over streaming data are to be reported continuously. Besides the accuracy (especially for multi-dimensional queries) and space consumption (for storage of statistics), which are important to the conventional estimation methods, the speed of estimation and updatability of estimators become just as important (as the accuracy and space) in the data stream environment.

In this paper, we develop a statistical selectivity estimation method based upon the empiric distribution estimation by the cosine series [12]. The cosine transform is known for its near-optimal energy compaction property and simple computation. Thus, the estimators are accurate and are easy and fast to derive. In addition, the estimators can be updated dynamically. Our experimental results confirm that the proposed estimators are accurate, efficient, and adaptive.

The rest of paper is organized as follows. In Sections 2 and 3, we introduce related work and define the terminology, respectively. In Sections 4 and 5, we lay down the groundwork for applying cosine transform to estimation of aggregate queries. In Section 6, we discuss the updatability of the proposed method. Section 7 compares the performance of the cosine transform with the wavelet method. Section 8 gives the conclusion.

## 2 Related Work

In this section we briefly review previous work on selectivity estimation and discuss their potentials for use in the data stream environment.

### 2.1 Sampling

Various sampling methods [13, 14, 21, 30] have been used to estimate resulting sizes of queries over conventional data. Sampling generally works very well for queries that post a constraint on a single attribute, called one-dimensional queries here; but the accuracy degrades drastically for queries with constraints posted on multiple attributes (or multi-dimensional queries), as demonstrated in the join operations [14].

Only recently, sampling has been adapted to data streams [8]. It is still an ongoing research on how to maintain a dynamic sample over data streams.

### 2.2 Histograms and Compression

Histograms [7, 18, 25, 26, 27] have been used successfully to capture the distributions of low-dimensional data. However, as the dimension increases, the number of histogram buckets can increase exponentially, rendering these methods prohibitively costly for multi-dimensional data.

To alleviate the size problem of histograms, wavelet transform has been used to compress histograms [5]. It was shown to yield favorable performance than histograms in point and range query size estimation [5, 23]. Martias et al. [24] has supplemented this method with a mechanism such that it can dynamically modify its coefficients in the face of updates. Recently, wavelets [5, 10] have been applied to data stream processing.

Discrete cosine transform (DCT) was used to approximate frequency distributions of histograms [20]. It is suitable for selection queries over discrete data.

In this study we use the cosine transform to approximate the distribution of data, not the histograms (as by DCT), and use it to estimate the selectivity of queries. The cosine transform is known to have a good energy preserving property and thus can approximate data distributions accurately. It also has a simple, efficient, and dynamic update method. We choose to approximate the cumulative distribution of data because it is generally smoother than the data density function and thus higher accuracy can be expected. The method can be applied to both discrete and continuous attribute domains. It is noted that cosine transform always represents the best limit of DCT;

that is, if the partition of the underlying histogram is infinitely fine, then DCT can have the same performance as the cosine transform.

### 3 Notations

#### 3.1 Range Queries

Let  $T$  be a relation with numerical attributes  $X_1, \dots, X_d$ . We can view  $X=(X_1, \dots, X_d)$  as a d-variate random variable. For each  $1 \leq i \leq d$ , denoted by  $x_i$  is a value of the attribute  $X_i$ . Let  $R$  be the set of all real numbers and  $R^d$  be the set of all d-dimensional real vectors. If  $x=(x_1, \dots, x_d) \in R^d$  and  $y=(y_1, \dots, y_d) \in R^d$ , we say  $x \leq y$  if  $x_i \leq y_i$  for all  $i=1, \dots, d$ .

Suppose  $[l, r] \subset R$ ,  $a=(a_1, \dots, a_d) \in [l, r]^d$ ,  $b=(b_1, \dots, b_d) \in [l, r]^d$ , and  $a_i \leq b_i$  for all  $i=1, \dots, d$ . Let  $\{X_{i_1}, \dots, X_{i_k}\}$  be a set of attributes on which range constraints, such as  $a_i < X_i \leq b_i$ ,  $a_i \leq X_i \leq b_i$ ,  $a_i \leq X_i < b_i$ ,  $a_i < X_i < b_i$ , are posted in the queries. For simplicity, we will restrict the constraints to  $a_i < X_i \leq b_i$  for now, but will extend to other forms later. So, a range query  $Q(a, b)$  denotes a query with attributes  $a_i < X_i \leq b_i$ , for all  $1 \leq i \leq d$ . By denoting  $a_i = l$  and  $b_i = r$  for  $i \notin \{i_1, \dots, i_k\}$ , the range query can be rewritten as  $a_i < X_i \leq b_i$ , for all  $1 \leq i \leq d$ .

To simplify the notation and algorithm implementation, attribute values are normalized to a predetermined domain  $[l, r]$ . Let  $\max X_i$  and  $\min X_i$  be the maximal and minimal values of attribute  $X_i$  of relation  $T$ , respectively. Then, each value  $x_i$  of  $X_i$  can be normalized as follows:

$$x_i^z = \frac{x_i - \min X_i}{\max X_i - \min X_i} (r - l) + l, \quad \min X_i < x_i < \max X_i \quad (3.1)$$

From now on, we shall assume all attribute values are so normalized and shall not distinguish  $x_i$  from  $x_i^z$ , unless otherwise stated. Furthermore, we will set the normalized domain to  $[0, 1]$ , that is,  $l=0$ ,  $r=1$  for simplicity and uniformity.

#### 3.2 Empiric Distribution and Selectivity

Consider a random variable  $X$  that has a cumulative distribution function  $F$ . If  $F$  is known, the probability of  $X$  falling in the range  $(a, b]$  is

$$P\{a < X \leq b\} = F(b) - F(a). \quad (3.2)$$

For simplicity, we shall use the term “distribution” for “cumulative distribution” from now on. The empiric distribution is exactly the data distribution of the relation, without losing any information. If the data distribution is continuous, error rates can be large if the data distribution is represented by a histogram, which captures only the frequencies, not the distribution. While the empiric distribution may look like a “cumulative” histogram, it is a function rather than a collection of cumulative frequencies.

Considering a relation as a sample from  $R^d$ , each tuple of the relation can be viewed as an observation. Given a collection of observation  $\{V_1, \dots, V_n\}$ , the empiric distribution function  $\widehat{F}(x)$  is defined by

$$\widehat{F}(x) = \frac{\text{number of observations} \leq x}{n}, \quad x \in [l, r]^d. \quad (3.3)$$

Given a sample from a random variable  $X$ , we can use the empirical distribution constructed from the sample to estimate the distribution of  $X$ . For example, suppose  $\{0.2, 0.34, 0.45, 0.52, 0.63, 0.75, 0.8\}$  is a sample from a random variable  $X$ , we can estimate the probability  $P\{X \leq 0.48\}$  as the ratio of the number of elements that are less than or equal to 0.48 (i.e., 3) to the sample size (i.e., 7), that is  $3/7$ .

In this paper, our goal is to find an approximation, denoted by  $\widehat{F}_m(X)$ , to the empiric distribution  $\widehat{F}(X)$ . The approximation shall use much less space and yet with little information loss.

In order to quantify the query result size, we adopt the terms instance selectivity and distribution selectivity from [4]. The instance selectivity (probability)  $\sigma(a, b)$  of a range query  $Q(a, b)$  is given by the number of resulting tuples divided by the total number of tuples in  $T$ , whereas the distribution selectivity (probability)  $\widehat{\sigma}_m(a, b)$  is the probability that a tuple is in the hyper rectangle  $(a, b]$  with respect to the probability distribution associated with  $\widehat{F}_m(X)$  (which will be discussed later). Our goal is to use the distribution selectivity  $\widehat{\sigma}_m(a, b)$  to estimate the instance selectivity  $\sigma(a, b)$ , the true selectivity.

## 4 Empirical Distribution Estimation via Cosine Series

In this section, we discuss how to derive an estimator of the empiric distribution.

In the one-attribute or one-dimensional case,  $\widehat{F}(x)$  is defined on the real numbers  $\mathbb{R}$ . A naïve way to store  $\widehat{F}(x)$  is to equally partition the interval  $[0,1]$  into  $k$  sub-intervals  $0 \leq a_1 \leq a_2 \dots \leq a_k \leq 1$  and then store the cumulative frequencies  $\{\widehat{F}(a_1), \widehat{F}(a_2), \dots, \widehat{F}(a_k)\}$ . If  $x \notin \{a_1, a_2, \dots, a_k\}$ , we can compute  $\widehat{F}(x)$  by linear interpolation. To achieve higher accuracy, we can choose finer partitions.

While the empirical function  $\widehat{F}(x_1, \dots, x_d)$  describes the exact distribution of tuples, the storage of such a function could be substantial, especially when the number of attributes ( $d$ ) and the domain sizes of the attributes are large. To save storage space, we opt to approximate the function by a cosine series.

Let the cosine series be denoted as  $\phi_i(x)$ ,  $i \geq 0$ ,

$$\phi_i(x) = \begin{cases} 1, & i = 0; \\ \sqrt{2} \cos i\pi x, & i > 0; \end{cases}$$

That is,  $\{1, \sqrt{2} \cos \pi x, \sqrt{2} \cos 2\pi x, \dots, \sqrt{2} \cos i\pi x, \dots\}$ . Let  $\Phi_i(x) = \int_x^1 \phi_i(u) du$ . That is,

$$\Phi_i(x) = \begin{cases} 1-x, & i = 0; \\ -\sqrt{2} \sin i\pi x / i\pi, & i > 0; \end{cases}$$

Given a relation  $T$  with  $d$  attributes and  $n$  tuples  $\{V_1, V_2, \dots, V_n\}$ , the empirical distribution  $\widehat{F}(x_1, \dots, x_d)$  of the attribute values can be approximated by a cosine series with  $m^d$  coefficients  $\widehat{\beta}_{i_1, \dots, i_d}$   $0 \leq i_1, \dots, i_d \leq m-1$ , as

$$\widehat{F}(x_1, \dots, x_d) \approx \widehat{F}_m(x_1, \dots, x_d) = \sum_{i_1=0}^{m-1} \dots \sum_{i_d=0}^{m-1} \widehat{\beta}_{i_1, \dots, i_d} \prod_{j=1}^d \phi_{i_j}(x_j), \quad (4.1)$$

The  $\widehat{\beta}_{i_1, \dots, i_d}$ 's are derived as

$$\widehat{\beta}_{i_1, \dots, i_d} = \frac{1}{n} \sum_{k=1}^n \left( \prod_{j=1}^d \Phi_{i_j}(V_{kj}) \right) \quad (4.2)$$

where  $V_{kj}$  is the  $j^{\text{th}}$  attribute of tuple  $V_k$ ,  $1 \leq k \leq n$ . In general, the larger the  $m$  value, the better the approximation. It is noted that only the  $m^d$  coefficients (real numbers) need to be stored for the approximate frequency function. Figure 4.1 summarizes the derivation of the coefficients of  $\widehat{F}_m(x)$ , i.e., Eq (4.2), in the following CosCoef algorithm.

```

Algorithm CosCoef( $m, V[n]$ )
  Input: an integer  $m$  and  $n$  tuples  $V_1, \dots, V_n$ , where  $V_k = (V_{k1}, \dots, V_{kd})$ ,  $1 \leq k \leq n$ ;
  Output: the coefficients  $(\beta[0, \dots, 0], \dots, \beta[m-1, \dots, m-1])$  of the cosine estimator  $\hat{F}_m(x)$ .
begin
1   $\beta[i_1, \dots, i_d] \leftarrow 0$  for all multiple indexes  $(i_1, \dots, i_d)$ ,  $0 \leq i_j \leq m-1, j = 1, \dots, d$ ;
2  for  $k \leftarrow 1$  to  $n$ 
3    do
4      for  $i_1 \leftarrow 0$  to  $m-1$ 
5        .....
6      for  $i_d \leftarrow 0$  to  $m-1$ 
7        do
8           $\beta[i_1, \dots, i_d] \leftarrow \beta[i_1, \dots, i_d] + \prod_{j=1}^d \Phi_{ij}(V_{kj})$ ;
9  for  $i_1 \leftarrow 0$  to  $m-1$ 
10 .....
11 for  $i_d \leftarrow 0$  to  $m-1$ 
12 do
13    $\beta[i_1, \dots, i_d] \leftarrow \beta[i_1, \dots, i_d]/n$ ;
end

```

**Figure 4.1** Coefficients of the Estimator

**Example 4.1:** Consider a one-attribute relation with 6 tuples  $\{0.33, 0.32, 0.12, 0.66, 0.90, 0.80\}$ . The cosine transform of this distribution is derived as follows.

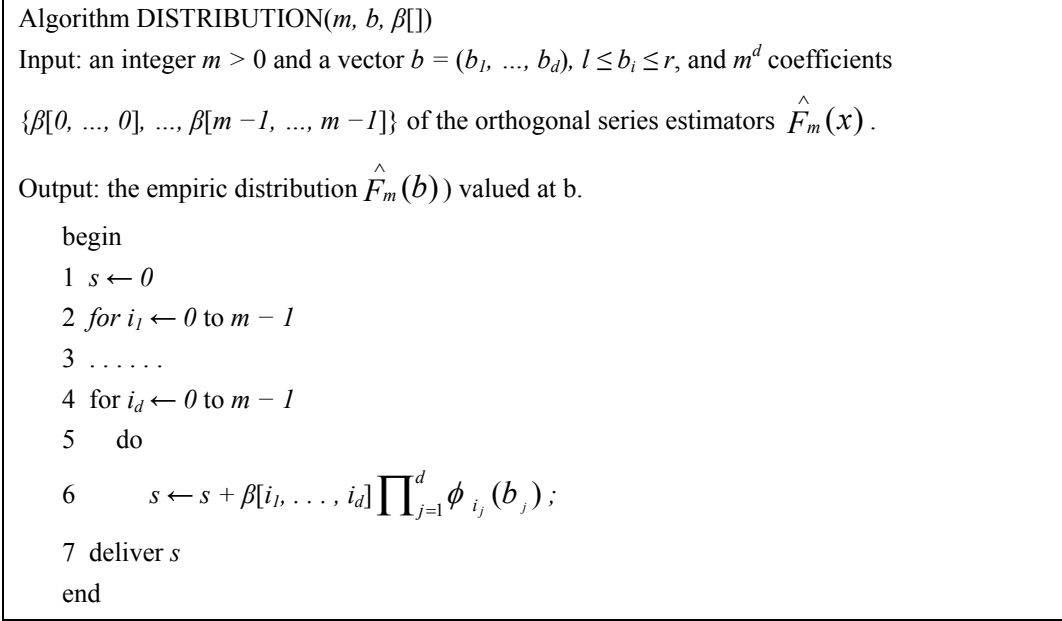
$$\hat{\beta}_0 = \frac{1}{6} \sum_{j=1}^6 \Phi_0(x_j) = \frac{1}{6} \sum_{j=1}^6 (1 - x_j) = 0.4783,$$

$$\hat{\beta}_1 = \frac{1}{6} \sum_{j=1}^6 \Phi_1(x_j) = \frac{1}{6} \sum_{j=1}^6 \left[ -\frac{\sqrt{2}}{\pi} \sin \pi x_j \right] = -0.2886,$$

Thus, the estimator of the empiric distribution with 2 coefficients is

$$\hat{F}_2(x) = 0.4783 - 0.2886 \sqrt{2} \cos \pi x$$

The computation of  $\hat{F}_m(x)$  at a specific point  $b$  is shown in Figure 4.2.



**Figure 4.2** Compute the Empiric distribution  $\hat{F}_m(x)$  at  $b$

### 5 Selectivity Estimation for Range Queries

In this section, we elaborate on the selectivity estimation of range queries using the approximate empiric distributions derived in the previous section.

Let us illustrate our idea through a 1-dimensional case. Suppose  $X$  is a random variable such that  $X \subset [0, 1]$  with a cumulative distribution function  $F(x)$ . The probability that  $a < X \leq b$  is

$$P\{a < X \leq b\} = F(b) - F(a). \tag{5.1}$$

The instance selectivity (probability)  $\sigma(a, b)$  is

$$\sigma(a, b) = \hat{F}(b) - \hat{F}(a) \tag{5.2}$$

and the distribution selectivity (probability) is

$$\hat{\sigma}_m(a, b) = (\hat{F}_m(b) - \hat{F}_m(a)) / (\hat{F}_m(1) - \hat{F}_m(0)) \tag{5.3}$$

with a pre-determined positive integer  $m$ . Note that  $\hat{F}_m(1)$  may not be 1 and  $\hat{F}_m(1) - \hat{F}_m(0)$  may not be 1 either. In order to guarantee  $\hat{\sigma}_m(0, 1) = 1$  (and have a better approximation), we adjust the estimation  $\hat{F}_m(b) - \hat{F}_m(a)$  by dividing it by  $\hat{F}_m(1) - \hat{F}_m(0)$  in the Eq. (5.3).

For simplicity, we denote  $\hat{F}_m(b) - \hat{F}_m(a)$  as  $P_m\{(a, b)\}$ . The distribution selectivity (probability) of a range query is now rewritten as

$$\hat{\sigma}_m(a, b) = P_m\{(a, b)\} / P_m\{(0, 1)\} \quad (5.4)$$

Now, let us extend  $X$  to a  $d$ -variate random vector. Let  $a = (a_1, \dots, a_d) \in [0, 1]^d$ ,  $b = (b_1, \dots, b_d) \in [0, 1]^d$ , and  $a < b$ . A vertex of the hyperinterval  $(a, b]$

$$(a, b] = \{x = (x_1, \dots, x_d) \in [0, 1]^d : a_1 < x_1 \leq b_1, \dots, a_d < x_d \leq b_d\} \quad (5.5)$$

is denoted as  $v = (v_1, \dots, v_d) \in [0, 1]^d$  with  $v_i \in \{a_i, b_i\}$  for  $i=1, \dots, d$ . Let  $\Delta_k(a, b)$  be the set of all vertices  $v$  with  $v_i = a_i$  for exactly  $k$  coordinates and  $v_j = b_j$  for the remaining coordinates.

Then,

$$P\{a_i < X_i \leq b_i, \quad \forall 1 \leq i \leq d\} = \sum_{k=0}^d (-1)^k \sum_{v \in \Delta_k(a, b)} F(v). \quad (5.6)$$

**Example 5.1.** Suppose  $d=2$ ,  $a=(a_1, a_2) \in [0, 1]^2$ ,  $b=(b_1, b_2) \in [0, 1]^2$ ,  $a_1 \leq b_1, a_2 \leq b_2$ . Then, by Eq. (5.6)

$$P\{a_i < X_i \leq b_i, \quad \forall 1 \leq i \leq 2\} = F(b_1, b_2) + F(a_1, a_2) - F(a_1, b_2) - F(b_1, a_2).$$

Let  $P_m\{(a, b)\} = \sum_{k=0}^d (-1)^k \sum_{v \in \Delta_k(a, b)} \hat{F}_m(v)$ . Since it is possible that  $P_m\{(a, b)\}$  has a negative value, especially when the selectivity (probability) is small, we set  $P_m\{(a, b)\}$  to 0 if the derived value is negative.

Now, we extend the above discussion to range queries with other types of constraints. Suppose a range query has the form  $a_i \leq X_i \leq b_i$  for  $i \in \{i_1, \dots, i_k\}$ , and  $a_i < X_i \leq b_i$  for others. We can approximate this type of general range queries as follows. First, choose a small vector  $\delta = (\delta_1, \dots, \delta_d)$  with  $\delta_i > 0$  for  $i \in \{i_1, \dots, i_k\}$ , and  $\delta_i = 0$  otherwise. Then let  $a_i' = a_i - \delta_i$  for all  $i$  and  $a' = (a_1', \dots, a_d')$ . The distribution selectivity is given by

$$\hat{\sigma}_m(a', b) = P_m\{(a', b)\} / P_m\{(0, 1)\} \quad (5.7)$$

We can specify  $\delta_i$  as zero or a small positive number, such as 0.0000 ... 01. Other range queries with  $a_i \leq X_i < b_i$  and/or  $a_i < X_i < b_i$  can all be handled in the same way.

**Example 5.2:** We use the one-attribute relation in Example 4.1 to show how to compute the selectivity. The query “Select \* from Salary where Salary  $\leq$  0.4 and Salary  $>$  0.2” returns 2 tuples. So the instance selectivity is

$$\sigma(0.2, 0.4) = 2/6 = 1/3.$$

While the distribution selectivity is given by

$$\hat{\sigma}_2(0.2, 0.4) = P_2\{0.2 < t \leq 0.4\} = \hat{F}_2(0.4) - \hat{F}_2(0.2) = 0.25.$$

The relative error is 25%.

Figure 5.1 summarizes the computation for the distribution selectivity  $P_m\{(a, b)\}$ , i.e., Eq (5.6).

```

Algorithm Selectivity (m, a, b,  $\beta$ [])
Input: an integer  $m > 0$ , two vectors  $a = (a_1, \dots, a_d)$ ,  $b = (b_1, \dots, b_d)$ ,  $1 \leq a_i \leq b_i \leq r$ , and  $m^d$ 
coefficients  $\{\beta[0, \dots, 0], \dots, \beta[m-1, \dots, m-1]\}$  of the orthogonal series estimators  $\hat{F}_m(x)$ .
Output: the probability estimation  $P_m(a, b)$ .
begin
1  prob  $\leftarrow$  0;
2  k  $\leftarrow$  1;
3  for i  $\leftarrow$  0 to  $2^d - 1$ 
4    do
5      for j  $\leftarrow$  0 to d - 1
6        do
7          if i MOD  $2^j = 0$ 
8            then v[j] = a[j];
9            k  $\leftarrow$  (-k);
10         else
11           v[j] = b[j];
12         prob  $\leftarrow$  prob + k  $\times$  DISTRIBUTION(m, v,  $\beta$ );
13  if prob  $>$  0
14    deliver prob;
15  else deliver 0;
End

```

**Figure 5.1** Computing the Distribution Selectivity

In general, lower frequency terms in cosine series contribute more to the estimation than higher frequency ones. Therefore, it is possible to store only the lower frequency coefficients without much information loss. To filter out higher frequencies, a technique, called Triangle Sampling [20] can be applied. It stores only those coefficients whose indexes satisfy  $i_1 + \dots + i_d \leq m-1$ . Thus, the number of the coefficients finally stored is  $C(m+d-1, d) \approx m^d / d!$ , which is much less than  $m^d$ . Note that the indexes ( $i_1, \dots, i_d$ ) of the coefficients need not be stored because they are unique and can be derived on the fly. For example, consider a 2-dimensional case ( $d = 2$ ) and  $m$  has been set to 3. Then, there will be  $m^d (=3^2)$  coefficients in the approximation function, denoted as  $C_{i,j}$ ,  $0 \leq i, j \leq m-1$ . By the triangle sampling, only 6 ( $=C(m+d-1, d)$ ) of them that satisfies the condition:  $i_1 + i_2 \leq m-1 = 2$ , are kept. They are:  $C_{0,0}, C_{0,1}, C_{0,2}, C_{1,0}, C_{2,0}, C_{1,1}$ . We will incorporate this technique in our implementation.

## 6 Dynamic Maintenance of the Estimator

As observed from Eq. (4.2), each coefficient  $\hat{\beta}_{i_1, \dots, i_d}$  of the transform is just the average of sum of the products of basis functions on the tuples. Therefore, for insertion or deletion of a tuple, we can just compute the ‘‘contribution’’ of the tuple to the transform separately and then combine them with the old coefficients. That is, for insertion of a new tuple  $t = (x_1, x_2, \dots, x_d)$ ,  $\hat{\beta}_{i_1, \dots, i_d}$  is update as

$$\hat{\beta}_{i_1, \dots, i_d} = \frac{n}{n+1} \hat{\beta}_{i_1, \dots, i_d} + \frac{\prod_{j=1}^d \Phi_{i_j}(x_j)}{n+1}. \quad (6.1)$$

Similarly, for deletion of a tuple  $t = (x_1, x_2, \dots, x_d)$ , it is updated as

$$\hat{\beta}_{i_1, \dots, i_d} = \frac{n}{n-1} \hat{\beta}_{i_1, \dots, i_d} - \frac{\prod_{j=1}^d \Phi_{i_j}(x_j)}{n-1}. \quad (6.2)$$

Let  $m$  be the number of coefficients for the estimator. Then the complexity for an update is  $O(m)$ .

Coefficients can be updated easily and dynamically. This property makes the cosine transform very suitable for data stream environments, where tuples continuously flow in. The updates of the coefficients can be performed on the fly as well as in batch. In addition, the computation workload can be easily distributed among processors as the ‘‘contributions’’ of tuples can be computed separately.

## 7 Experimental Results

In this section, we report the experimental results of selectivity estimation of queries using the cosine and wavelet methods.

### 7.1 Experiment Setup

We have implemented both methods in C++ and compiled them with GNU C/C++ Compiler V3.2.3. The test platform is Redhat Linux Enterprise 4 running on Dell Precision 360 workstation with 3.3 GHZ CPU and 1GB RAM.

Experiments are run on both synthetic and real-life datasets. The purpose of using synthetic data is to study the methods in relation to different characteristics of data in a controlled environment. Hopefully, the results can provide a general idea about the behaviors of the methods. The synthetic relations are generated following the TPC-D benchmark [28] with attribute values distributed Zipfianly [31]. We generate distributions with two different  $z$  values, 0.5, and 1.0, which represent, roughly speaking, a slightly skewed and skewed distribution, respectively. The domain size of each attribute is 1,024 and each relation has  $10^6$  ( $= N$ ) tuples.

We have also used a real-life dataset: the current population survey, the income and program participation survey from the Bureau of Census, for our experiments. We select the data for a period of three-months, from January to March 2004, for our experiments. The dataset has around 140,000 tuples for each month. The attributes: Age, Education and State, whose ranges are [1, 99], [1, 46] and [1, 99], respectively, are used in the experiments.

### 7.2 Estimation Accuracy

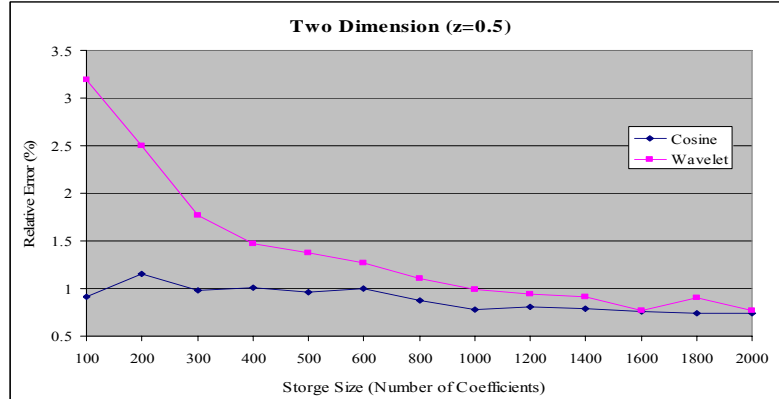
We ran 100 queries with randomly chosen ranges on selected attribute domains. The accuracy of selectivity estimation is measured by the average relative errors.

#### 7.2.1 Performance on Synthetic Datasets

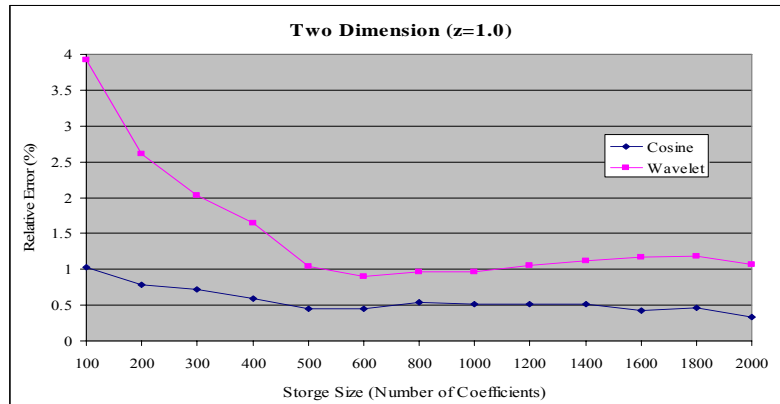
Figures 7.1 and 7.2 show the estimation results with range queries on two-dimensional data with Zipf parameters, 0.5 and 1.0, respectively. As shown in the figures, in general, the greater the number of coefficients used, the better the results of both techniques.

As shown in Figure 7.1, the cosine method performed better than the wavelet for the same number of coefficients for the slightly skewed distribution  $z=0.5$ . The wavelet generated average

errors ranging from 3.93% for 100 coefficients to 1.07% for 2,000 coefficients, while our approach generated from 1.02 % to 0.33% for the same number of coefficient, respectively.



**Figure 7.1** Two-dimensional Queries,  $z=0.5$



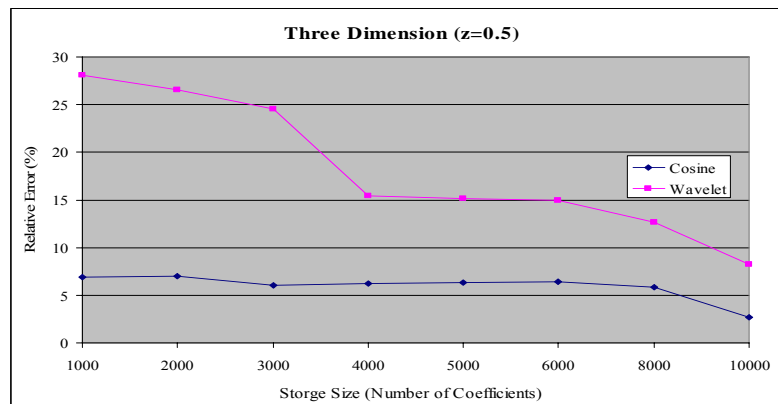
**Figure 7.2** Two-dimensional Queries,  $z=1.0$

Notice that with only 100 coefficients, our estimates are already very accurate (around 1% error) in both cases. The accuracy does not improve much as the number of coefficients increases. This demonstrates the good energy preserving property of the cosine transform, that is, energy is mostly preserved in lower frequency terms, as mentioned earlier in Section 5.

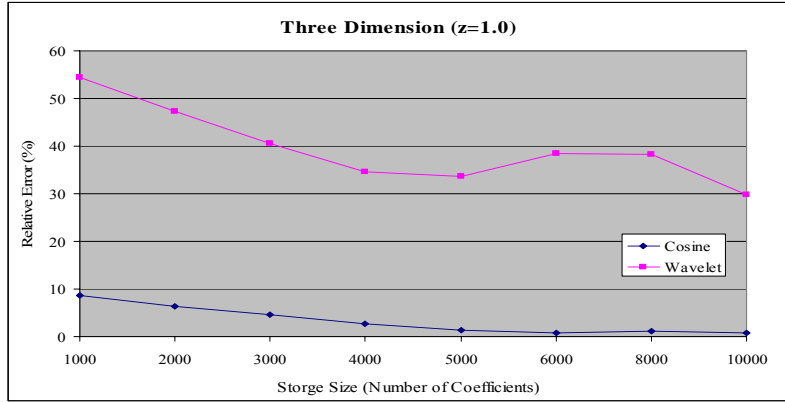
As distributions become more skewed (i.e.,  $z=1.0$ ), it becomes more difficult to capture the sharp changes in frequency and thus estimation becomes harder. Comparing Figures 7.1 and 7.2, it is observed for the same number of coefficients, the errors are larger for both methods in the latter. Our method seems to perform better relatively in the more skewed case ( $z=1.0$ ) than in the smoother case ( $z=0.5$ ).

Figures 7.3 and 7.4 show the results of range queries with constrains on three attributes with Zipf parameters 0.5 and 1.0, respectively. In general, the higher the dimension, the greater the number of coefficients needed to achieve a desired accuracy. This is mainly due to there being more frequency values to be approximated (or compressed) in higher dimensional spaces. Again, our method performed better than the wavelet for the same number of coefficients. In Figure 7.3, wavelet generated average errors ranging from 28.04 % for 1,000 coefficients to 8.25% for 10,000 coefficients, while ours from 6.9 % to 2.67% for the same number of coefficients. Wavelet’s errors are about 4 times larger than ours.

As distributions become more skewed, the errors become larger like in the 2-dimensional cases. For example, at  $z=1.0$ , the wavelet generated average errors ranging from 54.45% for 1,000 coefficients to 29.88% for 10,000 coefficients, while our approach generated from 8.68% to 0.81% for the same number of coefficients. The wavelet’s errors are about 6 to 37 times larger than ours.



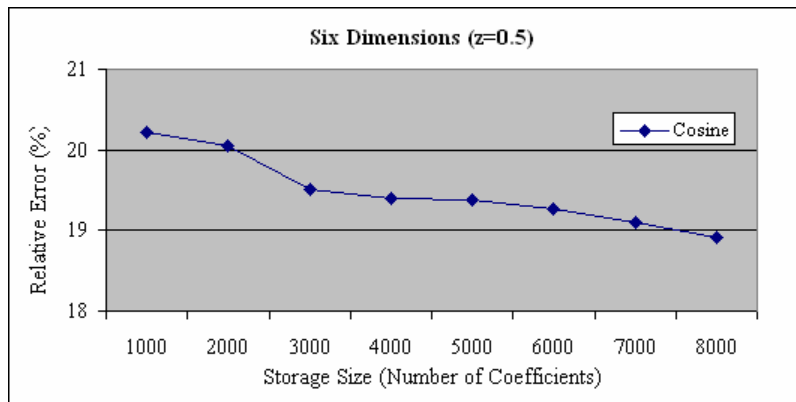
**Figure 7.3** Three-dimensional Queries,  $z=0.5$



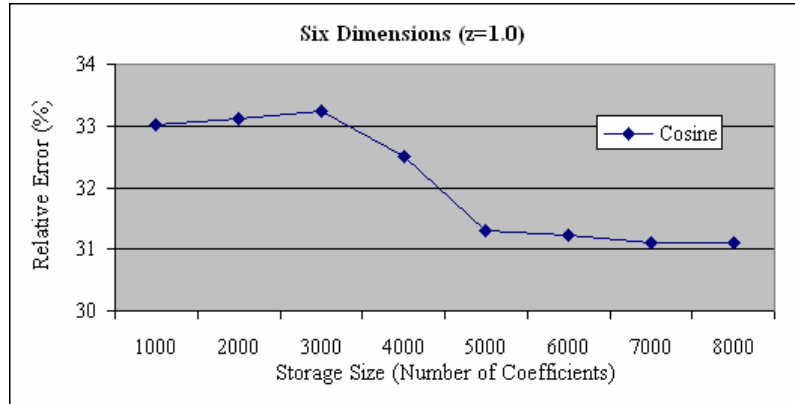
**Figure 7.4** Three-dimensional Queries,  $z=1.0$

In Figures 7.5 and 7.6, we show the results of six-dimensional queries. With each dimension partitioned into 1024 regions, the histogram has  $1024^6 (=2^{60})$  buckets, which is too large to be constructed. We have no choice but to partition each dimension into a much smaller number of regions, here 16 regions, which results in a  $16^6 (=16 \text{ million})$ -bucket histogram. The wavelet generated large errors (e.g.,  $> 100\%$ ) for small numbers of coefficients (e.g., 1,000, 2,000, etc). Even for the largest number of coefficients we tested, i.e., 8,000 coefficients, the errors are still very large, for example, 49.5% for  $z=0.5$  and 59.3% for  $z=1.0$ . Due to the large errors of wavelet, we present only the results of cosine series in the following.

As observed, it requires a much greater number of coefficients to achieve a desired precision in high dimensional spaces as there are too many frequency values to be approximated.



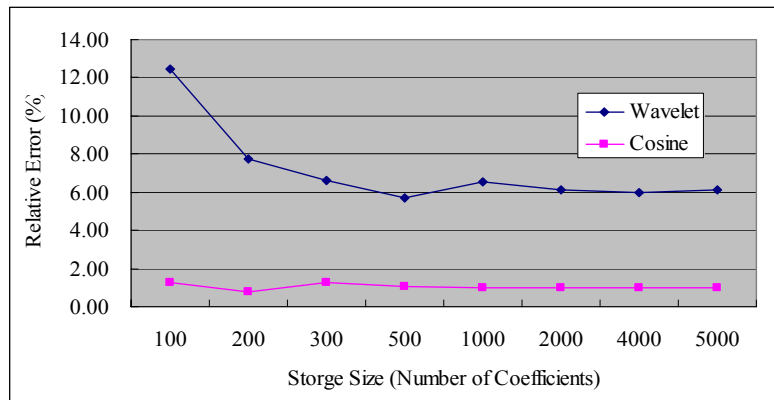
**Figure 7.5** Six Dimensional Queries,  $z=0.5$



**Figure 7.6** Six Dimensional Queries,  $z=1.0$

### 7.2.2 A Real Dataset

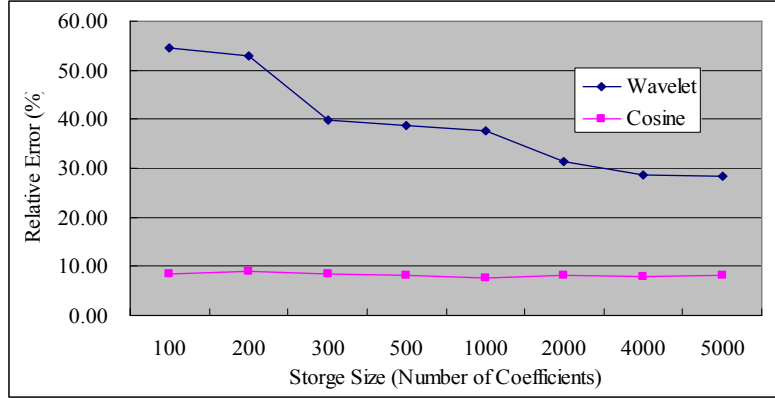
Figures 7.5 and 7.6 show the results on the real dataset.



**Figure 7.7** Real Dataset: Two-dimension Queries

As in the synthetic experiments, our method performs better than the wavelet. For example, with only 100 coefficients, as shown in Figure 7.7, the errors of the wavelet and cosine methods are 12.45% and 1.68%, respectively.

For three-dimensional queries, as shown in Figure 7.8, with 100 coefficients, our error is already below 10% while the wavelet still has an error as high as 54%.



**Figure 7.8** Real Dataset: Three-dimension Queries

### 7.3 Update and Estimation Speeds

It is important that selectivity estimates can be derived fast in a data stream environment. It is also important that estimators can be updated fast enough to accommodate the continuously arriving data. In the following, we compare the update and estimation speeds. As mentioned earlier, the higher the number of dimensions, the greater the number of coefficients is required for an estimator to achieve an acceptable accuracy. Therefore, we assume that the estimators have 400, 3,000, 8,000 coefficients for 2-dimensional 3-dimensional, and 6-dimensional cases, respectively, which we believe would generally be large enough to yield reasonable accuracy.

Let  $m$  be the number of coefficients used in the estimators. It takes  $O(m)$  time to update a cosine estimator, as shown in Eq. (6.1). On the other hand, wavelet takes  $O(\log H)$  time to update the coefficients, where  $H$  is the size of the underlying histogram, recalling that wavelet is a histogram-based method. Note that the size of histogram generally increases exponentially with the number of dimensions ( $d$ ) of the histogram, i.e.,  $H=|D|^d$ , where  $|D|$  is the size of the attribute domain. As shown in Table 7.1, wavelet is faster in 2-dimensional case but slower in higher dimensional cases. This is mainly due to the dramatically increased underlying histogram size of the wavelet method as the dimension increases.

Table 7.1 Update Speed

Time (microseconds)	2- dimension	3- dimension	6- dimension
Wavelet	210	2,058	-
Cosine	72	389	1,321

The cosine method has a complexity of  $O(2^d m)$  for selectivity estimation, as demonstrated in Eqs. (3.2) and (5.5). The wavelet needs to reconstruct the portion of histogram covered by the range query and thus has a complexity of  $O(2^d H \log(H))$ . Consequently, the wavelet estimator can be very slow if the relevant portion of the histogram is large. Similar to the update speed, wavelet is faster in lower dimensions, e.g., two, but slower in higher dimensions.

Table 7.2 Estimation Speed

Time (microseconds)	2- dimension	3- dimension	6- dimension
Wavelet	4.7	6,906	-
Cosine	132	1,250	3,002

## 8 Conclusions

In this paper, we develop a nonparametric statistical selectivity estimation approach, which is based upon the empiric distribution estimation by the cosine series. First, we derive an estimator for the empiric distribution of attributes in a relation, and then use the empiric distribution estimator to compute the distribution selectivity. Our method is suitable for both continuous and discrete data.

The empiric distribution estimator can be stored easily and updated efficiently. The experimental results have shown that our approach produced much more accurate estimates than the wavelet method. The proposed method is well suited for on-line approximate aggregate query estimation over continuous data streams. It is simple, accurate, efficient, and adaptive.

## References

1. Acharya, S., Gibbons, P., Poosala, V., Ramaswamy, S., Join synopses for approximate query answering. In SIGMOD. ACM Press, pages 275-286, 1999.
2. Babu S., and Widom, J., Continuous queries over data streams. SIGMOD Record, 30(3): pages 109-120, 2001.
3. Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J.,. Models and issues in data stream systems. In PODS, 2002.
4. Blohsfeld, B., Korus, D., Seeger, B., A Comparison of Selectivity Estimators for Range Queries on Metric Attributes, ACM SIGMOD Conference, pp. 239 -295, 1999.
5. Bulut, A., and Singh, A., SWAT: Hierarchical stream summarization in large networks. In IEEE International Conference on Data Engineering, 2003, pp.303-324.
6. Chakrabarti, K., Garofalakis, M., Rastogi, R., and Shim, K., Approximate query processing using wavelets, VLDB conference, pp. 111-122, 2001.
7. Christodoulakis, S., Estimating Block Transfers and Join Sizes, ACM SIGMOD Conference, 40–54, 1983.
8. Gemulla, R., Lehner, W., and Haas, P., A Dip in the Reservoir: Maintaining Sample Synopses of Evolving Data Sets, VLDB Conference, 2006, pp. 595-606.
9. Gibbons, P., and Matias, Y., New sampling-based summary statistics for improving approximate query answers. In ACM SIGMOD, pp. 331-342. 1998.
10. Gilbert, A., Kotidis, Y., Muthukrishnan, S., and Strauss, M., Surfing Wavelets on Streams: One-pass Summaries for Approximate Aggregate Queries In Proc. of VLDB, pp.79-88, 2001.
11. Guha, S., Koudas, N., and Shim, K., Data-streams and histograms. In Proc. ACM Symp. Theory of Computing (STOC), pp. 471-475, 2001.
12. Hall, P., Orthogonal Series Distribution Function Estimation, with Applications, Journal of the Royal Statistical Society, Series B, Vol. 45, No. 1, 81–88, 1983.
13. Haas, P., and Swami, A., Sequential Sampling Procedures for Query Size Estimation, ACM SIGMOD Conference, pp. 341–350, 1992.
14. Hou, W., Ozsoyoglu, G., and Taneja, B., Statistical Estimator for Relational Algebra Expression, ACM SIGMOD Conference, pp. 278–287, 1988.
15. Ioannidis, Y., and Christodoulakis, S., Optimal Histograms for Limiting Worst-Case Error Propagation in the Size of Join Results. ACM Transactions on Database Systems, Vol. 18, No. 4, pp. 709-748, December 1993.
16. Ioannidis Y., and Poosala, V., Balancing Histogram Optimality and Practicality for Query Result Size Estimation. In Proceedings of the 1995 ACM SIGMOD Conference, pp. 233-244, March 1995.
17. Ioannidis, Y., and Poosala, V., Histogram-Based Approximation of Set-Valued Query Answers, Proc. of 25th VLDB Conference, pp. 174-185, 1999
18. Kooi, R., The OPTimization of Queries in Relational Database Systems, Ph. D. Thesis, Case Western University, 1980.
19. Koudas, N., and Srivastava, D., Data stream query processing: A tutorial, VLDB, 2003.

20. Lee, J., Kim, D., and Chung, C., Multi-dimensional Selectivity Estimation Using Compressed Histogram Information, ACM SIGMOD, pp. 205-214, 1999
21. Lipton, R., Naughton, J., and Schneider, D., Practical Selectivity Estimation through Adaptive Sampling, ACM SIGMOD Conference, pp. 1–12, 1990.
22. Ling, Y., and Sun, W., A Supplement to Sampling Based Methods for Query Size Estimation in a Database System, ACM SIGMOD RECORD, pp. 12–15, 1992.
23. Matias, Y., Vitter, J., and Wang, M., Wavelet-Based Histograms for Selectivity Estimation. In Proceedings of the ACM SIGMOD Conference, pp. 448-459, 1998.
24. Matias, Y., Vitter, J., and Wang, M., Dynamic Maintenance of Wavelet-Based Histograms. Proc 26th VLDB Conference, pp. 101-110, 2000.
25. Muralikrishna, M., and DeWitt, D., Equi-depth Histograms for Estimating Selectivity Factors for Multi-dimensional Queries, ACM SIGMOD Conference, pp. 28–36, 1988.
26. Poosala, V., and Ioannis, Y., Selectivity Estimation Without the Attribute Value Independence Assumption, 23rd VLDB Conference, pp. 486–495, 1997.
27. Poosala, V., Ioannidis, Y., Haas, P., and Shekita, E., Improved Histogram for Selectivity Estimation of Range Predicates, ACM SIGMOD Conference, pp. 294–305, 1996.
28. TPC benchmark D, decision support, 1995.
29. Vitter, J., and Wang, M., Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets, ACM SIGMOD, pp. 193-204, 1999.
30. Wu, Y., Agrawal, D., and Abadi, A., Applying the Golden Rule of Sampling for Query Estimation, ACM SIGMOD, pp. 449-460, 2001.
31. Zipf, G., Human Behavior and the Principle of Least Effort, Addison-Wesley, Reading, MA, 1949.