Query Estimation in Data Streams using Micro-clustering

A thesis submitted

in fulfilment of the requirements

for the degree of Doctor of Philosophy

in

Computer Science and Engineering

by

Sudhanshu Gupta

(Reg. No. 950803014)

Supervision

Dr. Deepak Garg

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT

Thapar University

Patiala

CERTIFICATE

Certified that the work contained in the thesis titled "Query Estimation in Data Streams using Micro-clustering", by Sudhanshu Gupta (Reg. No. 950803014), has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

October 2014

(Dr. Deepak Garg)
Dept. of Computer Science & Engg.
Thapar University
Patiala

Abstract

Advancement in technology has lead to availability of inexpensive electronic devices everywhere. These devices and various applications automatically generate a large amount of data which is increasing exponentially. The data can grow at a high rate of millions of data items per day for business and scientific applications. A large number of applications generate continuous, transient large stream of data. For example the applications that naturally generate data streams are financial tickers, log records or click-streams in web tracking and personalization, manufacturing processes, data feeds from sensor applications, sensor network, performance measurements in network monitoring and traffic management, call detail records in telecommunications, email messages.

The analysis of large amount of data generated by various applications can create a lot of opportunities. For example, analyzing data of patients to diagnose the cause of disease, to design marketing strategies, predicting investment strategies, analyzing customer behavior. We need efficient techniques to analyze and process these unbounded data streams for useful information. However conventional techniques may not be applicable for their analysis. The processing of data stream requires single pass processing with limited memory. A number of techniques have been proposed for analysis of data streams meeting rigid processing requirement. These methods use various synopsis techniques such as sampling, wavelets, sketch etc.

Micro-clustering is a synopsis technique used for clustering and classification of data stream. In this work we investigate how to estimate queries over large data streams using micro-clustering and cosine series. We store summary of data stream in micro-clusters and process clusters of data for estimating queries over streams. In order to assess the technique we conducted an experimental study. As the results of this study reveal, our technique outperform competitor method.

The proposed techniques deal with evolving nature of data stream and estimate queries over data streams successfully.

Acknowledgements

First of all, I would like to thank my PhD supervisor Dr. Deepak Garg, for his consistent support and guidance . He provided me with vision, encouragement and counseling necessary to proceed throughout the doctoral program. I am much indebted to him for his affection and generosity throughout the tenure of my research work. Thank you very much for the great support.

I am extremely grateful to the members of doctoral committee Dr. Anil Kumar Verma and Dr. Inderveer Channa for their encouragement, insightful comments and valuable suggestions during all these years. I am grateful to Prof. Seema Bawa and Dr. Maninder Singh, former Heads of Department of Computer Science and Engineering, Thapar University, Patiala for their constant support and motivation. I am also thankful to all the faculty of Computer Science and Engineering department for their support.

I am deeply indebted to my parents for their love, prayers, care and sacrifices for educating and preparing me for my future.
I express my thanks to my wife and children for their continuous support during the hard time without which I could not have been succeeded. My special thanks to my brother and sister-in-law for their constant encouragement.
I would specially like to thank fellow PhD students Sh. Kuldeep Sharma and Rattan Rana for their consistent support. I wish to thank my friends and colleagues for their support and wishes.

Finally I thank almighty GOD ,the source of knowledge, and wisdom. From him we owe all that we have ,all that we are. Thank you GOD.

Contents

List of Figures

List of Tables

Dedicated To my Parents

Chapter 1

Introduction

In this chapter, we introduce the main topic of this thesis, namely query estimation over data stream using micro-clustering. We start in Section 1.1 with discussion on the importance of data stream mining and how it can contribute to the society. Then we outline how the mining and analysis of data streams can gain advantage from query estimation. In the section 1.2 we introduce data stream mining and section 1.3 discusses briefly query estimation over data streams. In this work, we present solutions for query estimation over data streams. Section 1.4 summarizes the main contributions of this thesis. Section 1.5 concludes this chapter with an overview of the organization of this thesis.

1.1 Motivation

With the advancement in technology, inexpensive electronic devices are available everywhere. The amount of data generated and stored by these devices and other data intensive applications is increasing exponentially. The data can grow at a high rate of millions of data items per day for business and scientific applications. Every day Google handles around 200 million searches, and AT&T produces more

than 300 million call records. The volumes of data collected by satellites are hundreds of gigabytes per day. Several applications naturally generate data streams: financial tickers, sensor network, performance measurements in network monitoring and traffic management, log records or click-streams in web tracking and personalization, manufacturing processes, data feeds from sensor applications, call detail records in telecommunications, email messages, and others.

The analysis of large amount of data generated by various applications can create a lot of opportunities. For example, by analyzing data gathered from patients, we can gain a better understanding about the cause of certain critical events. We can analyze data to design marketing strategies and take business decisions. We can use banking data to determine who gets a loan, and predict investment strategies. Similarly supermarkets can analyze customer data to boost sales by offering various schemes to customers.

We need efficient and automated techniques to analyze and process exponentially growing data to extract useful information. These techniques should work efficiently and should use minimum resources. In this work we investigate how to estimate queries over large data streams.

1.2 Data Stream Mining

Data stream is an infinite sequence of data that arrives continuously and at a high rate. It is generated by various applications such as ATM transactions, sensor networks [1], web clicks, telephone calls, network monitoring etc. It is transient and volatile in nature as elements arrive unpredictably in large volumes, with sudden changes in their characteristics. We process data streams to find the useful information [2, 3, 4, 5, 6, 7, 8] for various applications. As data arrives at high speed, each element has to be processed essentially in real time to store or update summary information. The sequence of data is infinite and cannot be stored in the memory completely for processing. We can store only a small summary and

throw away the rest of the information. This means streaming algorithm should work with-in the amount of memory limit and time-per-item. For example the monitoring of credit card transaction data should instantly detect the fraud. Similarly monitoring of vital signs of patients any unusual activity, which may induce a life-threatening condition of the patient should trigger an alert. In stock market analysis can help a trader to identify arbitrage opportunities. Moreover, in many of these applications, the underlying data distribution changes over time. Thus the past data may become irrelevant for the current summary. Real-time processing and dealing with evolving data poses challenges in the data stream analysis and processing.

In case of static datasets, data mining algorithms can afford to read the input data several times. However the data stream algorithms scan the data only once and have to deal with evolving nature of data streams. The data stream processing time should be proportional to the size of the data stream and the algorithms should approximate answers within acceptable probabilistic guarantees. In this work, we delve more deeply into one important stream mining and analysis task, namely query estimation over data streams.

1.3 Query estimation over data streams

The assessment of result size of a query, when applied on a given data distribution is called query estimation. Selectivity estimation is to find the fraction of data values satisfying a predicate to the total number of data values in the data stream. Selectivity estimation over data streams has significant applications in data exploration and query optimization. It helps in analyzing data for useful information in many applications such as trend analysis, decision support system, fraud detection, quality and performance monitoring where selectivity is ascertained for selection, projection and join queries. For example in query optimization [9] it is essential to select the best query execution plan. The query optimizer selects the

plan with the lowest cost from various available plans. The cost of plan is determined by the size of input argument. We select the best query plan by estimating selectivity of each predicate. The contributions of this thesis mainly concern with range query estimations. Selectivity of range selection query is to find fraction of data points lies in a predicate range i.e. $Q(a \leq X \leq b)$ where $a \leq b$. The multi-dimension range queries are intersections of ranges, each range being defined on a single attribute. Solving such a query involves counting how many points fall in the intersection of the ranges i.e. $(a_1 \leq A_1 \leq b_1) \wedge \cdots \wedge (a_d \leq A_d \leq b_d)$.

Selectivity estimation methods over unbounded data streams need to have minimum computational cost with optimal accuracy. The evolving nature of data streams makes it difficult to predict the behavior of data stream distributions. Hence, the selectivity estimation methods are designed to work independent of distribution type and have to work dynamically under insertion and deletion of data. It is desirable that these methods work efficiently under updates with minimum memory requirement.

1.4 Contributions

We present a novel approach for range query estimation for evolving data streams using micro-clustering and cosine coefficients. It is designed to meet the rigid processing requirements for data streams.

- The basic idea of our approach is to maintain a constant number of micro-clusters while processing the stream. We store the summary statistics of data stream in terms for micro-clusters in a way that is appropriate for selectivity estimation. Every micro-cluster contains the summary information about the cluster using cosine coefficients.
- The online data stream processing algorithms scan the data only once to maintain the summary.

- We use energy compaction properties of cosine series to store and process the data efficiently.
- The cosine estimator and micro-clusters estimates the selectivity while dealing with evolving nature of data streams. We can extend the technique to find selectivity over historical data by storing micro-clusters periodically offline. The technique is local and selectivity is estimated only by considering relevant clusters.
- The technique incrementally maintains the summary statistics. It reflects dynamic data updates to the statistics immediately.
- To evaluate our method over data streams, we conducted an experimental study and examined the performance for a variety of data streams, including real-world streams.

1.5 Outline of the Thesis

We discuss the contributions of the work in the organization of this thesis. Chapter 2 discusses literature on topics related to the work. We present survey on synopsis techniques, clustering and various approaches used for selectivity estimation.

Chapter 3 discusses the range query estimation framework using micro-clustering and cosine series. We present the micro-clustering as a way of storing summary information and cosine series as a density estimator. We discuss the implementation of the method for the cases of multiple data streams and distributed data streams. Chapter 4 thoroughly discusses rectangular query estimation over data streams. We start with the discussion of our method for maintaining micro-clusters over data streams followed by query estimation using cosine coefficients. Chapter 5 presents the results of our experimental study, which evaluates the performance and accuracy of our method. Besides assessing our techniques, we

compared them with competitive techniques. Chapter 6 concludes this thesis with a summary of its salient results. Additionally, we discuss future research directions.

Chapter 2

Literature Survey

The ubiquitous data streams have attracted lot of research in recent times. A large number of techniques have been proposed for query processing and analyzing over data stream. The importance of selectivity estimation in large number of applications has lead to a number of techniques. We provide relevant sources for further investigations in related topics. This chapter provides an overview of approaches related to the main topics of this thesis. Section 2.1 covers the literature related to processing data streams. In Section 2.2, we discuss synopsis methods for data streams Section 2.3 surveys query estimation methods and their adaptation to data streams.

2.1 Data Streams Processing

A data stream consists of a sequence of data values arriving continuously at different timestamps. Each data value consists of d attributes. This unbounded sequence of data cannot be processed within the limited resources. It poses space and time constraints on the computation process of data streams.

A data stream can be unbounded in size and the amount of storage required for computing an exact answer to a query may grow without bound. Most of

the algorithms which use external memory for storing data sets larger than main memory does not suite to data stream applications. These algorithms do not support continuous queries and are quite slow for real-time response. We require continuous data stream model in the applications where we need timely responses to queries and we receive large volumes of data continually at a high rate over time. We continuously receive new data while the old data is being processed. The time taken for computation must be low, otherwise algorithm will not be able to keep pace with the fast data stream. For this reason, algorithms [10] that are able to confine themselves to main memory are suitable for data stream processing rather than those using external memory.

The most of the query processing and mining algorithms should be executed efficiently which is not easy to achieve with fast, continuous and evolving data streams. All the data stream algorithms are designed to deal with the following issues:

- The algorithm should be able to adapt to the fast speed of streaming data.
- The data stream should be processed in one-pass as it is not feasible to scan the contents more than once during the course of computation.
- The query estimation requires data to be resident in the memory for finding accurate results. This is not feasible in case of data streams as they require unbounded memory for processing.
- There is an evolution in data streams over the time due to the change in underlying data patterns. It is necessary to deal with this evolution to have accurate result.
- The algorithms need to maintain a tradeoff between the accuracy of the results and the time and space complexity. The space complexity should not be more than linear in the size of the stream. They should be designed to be logarithmic in the domain-size of the stream.

The most of real life applications have attributes with large domain size. Arasu et al.[11] concluded that without the knowledge of size of the input data streams, it is not possible to place a limit on the memory requirements for common join queries. It is not feasible to process data with in the limited memory to find exact answers for data stream queries. Hence, high qulity approximate answers to the data stream queries are acceptable.

These above mentioned issues are solved using statistical and computational approaches [12, 13, 14, 15, 16]. we examine only a subset of the whole data set or to transform the data to an smaller size data representation. We construct the synopsis data structures and statistics from the streams which can be useful for various applications. These synopsis techniques are sampling, histogram, sketch, wavelet etc. Some of these techniques maintain the summary information about data while others perform change of basis. Wavelet transforms, discrete cosine transform etc. are decomposition techniques and change the basis. We compute each value in the data stream as a combination of the new basis. Many coefficients in this combination may be close to zero and are approximated to zero without much loss in accuracy. Hence, the B largest coefficients for a given input are kept such that mean square error is optimized. We apply inverse transformation for computing query. The distribution of coefficients is important in deciding accuracy of results. These synopsis methods have been used for solving selectivity estimation problem. We require a different synopsis structure for query estimation from a synopsis structure used for data mining problems of classification [17, 18] and change detection [17, 19, 20, 21, 22]. Moreover, we construct the synopsis structure so as to have a wide applicability for a variety of problems.

Many approaches for the processing[23, 24, 25, 26, 27] and querying [28, 29, 30, 31, 32] of data streams have been developed in recent years. Babcock et al.[23] examine models and issues for data streams. They discuss the difficulties of query processing over data streams and the usefulness of data summarization techniques in approximating solutions to query processing. MAIDS [33] is a stream mining system, that mine dynamics and alarming incidents in data stream. It uses a

tilted time window framework and multi-resolution model for multi-dimensional analysis. It performs classification, frequent pattern mining[34], and clustering of data streams and visualization of stream mining. Gigascope [35, 36] is a stream database for network monitoring applications. It provides solution for the problem of blocking operators. StatStream [37] is another data stream system to monitor statistics of multiple data streams in real time. It maintains average, standard deviation and correlations of all the pairs of streams with respect to sliding windows using discrete Fourier transform. VEDAS [38] is a mobile and distributed data stream mining system. It monitors the health of vehicles and the behavior of their driver. Telegraph [39, 40] is another such system developed for continuous data processing.

Data streams evolve over the time and results in change in underlying data distribution. Due to this change a particular computation model may become stale and has to adjust with the evolution of the data stream. Techniques [19, 20, 21, 41, 42, 43] have been proposed to study such changes to have insight into emerging trends in data. In [21], authors discuss challenges in mining changes over data streams. [42] use decision trees to incorporate changes of a data stream using a demand-driven active data mining concept. They address the problem of unknown class labels of a data stream. Charu et al.[19] capture such changes using the concept of velocity density which measures the rate of change of data concentration at a given spatial location over a user-defined time horizon. Authors use velocity density estimation in order to pick the projections from high dimensional data in which the greatest changes in data characteristics have occurred. They estimate over spatial vectors and time using kernel density estimators and observe that the velocity density is a scaled difference of kernel density estimators over various slices of time. The value of velocity density at a point gives the local rate of change of the density in a related time frame. In [20] author assume that data stream elements are drawn from a probability distribution to examine the change in a data stream. The technique continuously examines two windows with a pair of sliding windows over the data stream and a

reference window. These windows are updated on change and it is tested whether two windows follow the same distribution.

We use a variety of queries for data stream mining in different applications. We perform queries on data streams either on historical data or on recent data via sliding window query processing. The complete history of the data stream may not be useful in a repetitive data mining application. The sliding windows are more useful for data streams, because only a recent history is relevant in most of the applications. However, some times we may like to query stream from varying lengths of the history. There are two important types of queries applied in data stream model. These queries over data stream can be continuous [44, 45] or one shot in nature. Continuous queries are performed at regular intervals, for example finding number of telephone calls every five minutes. The one-shot queries are applied once over a snapshot of the data stream and return answer to the user only once. On the other hand continuous queries are the more useful in case of data stream and the answer to a these queries are generated over the time and always applied on the stream data seen so far. We may store these query answers and update them as new data arrives. These query answers may be produced as data streams. The aggregation queries involve frequent changes to answers, while join queries may generate rapid and unbounded answers.

Another categorization of queries is between predefined queries and ad hoc queries. A predefined query is supplied in advance before any relevant data has arrived and these are continuous in nature. Sometimes scheduled one-time queries can be predefined. On the other hand, Ad hoc queries are issued online after the data streams have already begun. These queries can be either one-time queries or continuous queries. They are difficult to handle as they are not known in advance. We may apply queries on data generated on one site or several remote sites on the network e.g. who uses most bandwidth, hosts with similar requirements etc. We need to process various complex queries over spatial data streams [46, 47]. All these queries are combination of three basic types of operations i.e. selection, projection and join.

2.2 Clustering techniques in data stream

Clustering of data stream is very important in various applications and has been studied extensively [18, 48, 49, 50, 51, 52, 53, 54, 55, 56]. Clustering algorithms can be divided into two types. First type is hierarchical algorithms, which start with each point in as own cluster and clusters are combined based on their closeness. The second type of algorithm assigns points to clusters. In this case points are assigned to the cluster into which it best fits. Some of the algorithms assumes a Euclidean space and summarize a collection of points by their centroid.

BIRCH (balanced iterative reducing and clustering using hierarchies) [57] is an unsupervised hierarchal data clustering algorithm. It is designed to work over particularly large data-sets and it incrementally and dynamically cluster incoming data points. It is local and each clustering decision is made without scanning all data points and currently existing clusters. It exploits the fact that most of the data space is not uniformly occupied. The method organizes clusters features in a height balanced tree. The clustering feature represents the associated sub-cluster.

k-means clustering divides the input data instances into k clusters, so as to minimize metric relative to the centroids of the clusters. The algorithm first places k points into the data space that is being clustered as initial group of centroids. Then it assigns new data point to the group that has the closest centroid. Further it recalculates the positions of each of the k centroid by taking the average of the points assigned to it. The k-median algorithm maintains variance and medians of sliding windows over data streams. It uses memory sub-linear in the window size. The algorithm continuously maintains exponential histograms and uses them to estimate the value of the k-median objective function.

A number of methods for clustering using partitioning are available in the literature. These methods are based on k-means and k-medians techniques. They define clusters as a set of data points selected from the data. We assign these data points to their closest cluster. Contrary to the data stream requirements, these

methods may require more than one pass over the data to accurately estimate the clusters.

The STREAM algorithm [58] uses the k-medians clustering. The technique breaks the data stream into chunks with a size that fits into main memory. Each chunk have at most m data points, the value of m is decided based on the available memory. They use k-medians algorithm and pick a set of representatives from each chunk so as to assign each point to its closest representatives. Then they pick the representatives so as to minimize the Sum of squared Error of the assigned data points. In this way they process chunk to generate k medians. Similarly other chunks are processed to find k-optimal median.This method is not sensitive to the evolution of data.

The CluStream [48] provides a technique for clustering evolving data streams and allows us to trace the evolution of clusters over different time horizons. It provides significantly improved view at different times and allow authors to compute clusters over different time horizons in one pass. The technique can determine the clusters in past times, and can compare them to the present clusters. For this purpose it divides the clustering process into an on-line micro-clustering component and an off-line macro-clustering component. The on-line micro-clustering component has an efficient process for storing of summary statistics for a data stream. The off-line component uses these summary statistics to provide the user with a understanding of the clusters. The micro-clustering component maintains temporal statistics of the data stream at a given time. These micro-clusters are defined as an extension of the cluster feature vector and maintain statistical information about the data locality of the stream. They are quite useful in case of data streams due to their additivity property. The technique stores the stream snapshot at different times with respect to pyramidal time frame. This provides a balance between the storage requirements and it facilitates us to use summary statistics from different times in past. The macro-clustering component uses these past statistics to generate clusters for a user-specified time frame.

The main memory contains the current snapshot of micro-clusters. The macro-clustering algorithm apply the additive properties of the micro-clusters stored at snapshots tc and tch so as to find the higher level clusters in a time slice of length h. The historical snapshots are stored on the disk. The more number of snapshots are stored in recent time frame. The micro-clustering does on line data collection at a high level of granularity so that it can be effectively used by the off-line components to find statistics for different time horizon. The number of clusters maintained is determined by the amount of main memory available in order to store the micro-clusters. These micro-clusters change when ever a new data point arrives and maintain the current snapshot of clusters. At the same time any micro-clusters which were stored at a time in the past are not updated for long are deleted.

Whenever a new data point arrives, the micro-clusters are updated. Each data point is either put in a micro-cluster, or it forms a cluster of its own. The first choice is to absorb the data point into an existing micro-cluster, if the data point falls within the boundary of a micro-cluster. The data point is added to the micro-cluster using the additivity property and define the boundary of the micro-cluster as a factor of t of the RMS deviation of the data points from the centroid. If the data point does not lie within the boundary of the nearest micro-cluster, then a new micro-cluster is created containing this new data point. In order to create this new micro-cluster, we need memory space which is created by reducing the number of clusters. This is either done by deleting an old cluster or by merging two of the nearest clusters. The timestamp data stored in micro-clusters is useful for the calculation of various useful statistics such as mean and standard deviation of the arrival times of points in a given micro-cluster. The time of arrival of the $m/(2 * n)$ th percentile of the points in a cluster is computed by assuming that the time-stamps are distributed normally. This time-stamp gives the value of recency of a micro-cluster and is called as relevance stamp of a cluster. When the smallest such time stamp of any micro-cluster is below a given threshold then it is deleted so as to create space for a new micro-cluster

having newly arrived data point. When none of there is no such micro-clusters available for deletion, the two nearest micro-clusters are merged. The ClusTree method [59] allows the adaptation of the granularity of the cluster model to the stream speed. The density-based stream Clustering has been proposed in literature. These methods creates a density profile of the data for clustering purposes. We can use kernel density estimation to create a smooth density profile of the underlying data. These density connected clusters can have different shapes and sizes. The density based clustering does not fix the number of clusters and uses a threshold on the density to determine the connected regions. However it is difficult to design density-based algorithms which can be work in a one pass of the data with minimum computations. Density based clustering is done either using micro-clustering technique or divides the data space into grids to find the dense grids.

The DenStream method [41] combines micro-clustering approach with a density-estimation process for effective clustering. In order to determine dense regions of smaller granularity are defined as core micro-clusters. A core micro-cluster is a collection of data points with weight based on decay weighted function of last arrival time. The method does not put constraint on the number of micro-clusters but the radius of each micro-cluster is constrained. It defines two types of micro-clusters the core micro-cluster as p-micro-clusters and outlier micro-cluster as o-micro-clusters. When a data item arrives the technique look to assign it to a p-micro-cluster without violating the radius constraint otherwise look to assign it to an o-micro-cluster without violating the radius constraint. In case the new point does not fall in p-microcluser or o-micro-cluster, it creates a new o-micro-cluster containing this data point. As number of o-micro-clusters increases with time, some of these are converted to p-micro-clusters and others are deleted after some threshold of time.

The grid-based methods is another type of density-based streaming algorithms. In which we use grid structure to calculate the density at each point in the data. We divide the data into different cells along each dimension by discretization of

data into ranges along that dimension. Then we aggregate dense cells in the data so as to find the dense regions for the clustering. The D-Stream [60] is a grid based clustering algorithm for streaming data that try to determine fine grained regions of high density. All the grids have a size limits and grid cells may change between the states i.e. sparse,transitional or dense, while the number of grids depends the dimensionality of data. They identify and remove sporadic grids from the data and consolidate maintained dense grids into larger dense regions in the data. The [61] maintain multi-dimensional grid of the data stream to discover clusters from the density values. The grid-based methods divide data along the different dimensions to create grids. The granularity of the discretization is a important design choice at the start of the algorithm. The granularity may not be selected optimally as the data distribution of data stream is not known initially and may be different in different localities. Probabilistic streaming algorithms for clustering uses mixture modeling. They assume data to be generated by a mixture of known distributions. These methods may not work when the underlying data is evolving rapidly.

The HPSTREAM [62] is a micro-clustering based method that performs high dimensional projected stream clustering. The algorithms uses an incremental algorithm in which link a group of dimensions with each cluster. It computes distances of data points from clusters by considering dimension-specific clusters. The additional information in the form of a bit vector containing one bit for each dimension is included in a cluster. A grid-based algorithm for high dimensional projected clustering of data streams are applied to uncertain data. The StreamCluCD algorithm [63] is a one-pass algorithm for clustering categorical data. It maintains the frequency counts of the attribute values in each cluster. In various application data stream generated contain data attribute with large domain. In network applications, IP-address attribute is drawn over millions of possibilities. In credit-card transactions can be drawn from a large number of different possibilities. Similarly supermarket transactions are often drawn from a millions of possibilities. These large domain attributes are difficult to process due to various

restrictions and high computation. In order to deal with these massive-domain cases the technique stores the frequency statistics of a large number of possible attribute values.

A sketch based technique [64] has been proposed to keep track of the intermediate statistics of the underlying clusters. It is based on count-min sketch [65] for the clustering of large domain data streams. It uses a hashing approach to keep track of the attribute-value statistics in the data. Whenever a new element of the data stream arrives, it applies a group of hash functions to map onto a number in range $[0 \ldots h1]$ and increment the count of each cells by one. The technique assigns every incoming data points to the nearest cluster. It uses sketch table to increment the frequency counts for the different attribute values in the cluster and measure the similarity between incoming data points and centroid of a cluster by computing of the dot-product function. The frequency sketches of the records are maintained and are assigned to the cluster. The algorithm maintains a sketch table having the counts of the values in the incoming data points. For each incoming data item, it will update the counts of each of the cluster-specific hash tables. In many applications such as sensor networks, a large volumes of data are collected at the different sensors. The clustering of distributed datastream require transmitting of all of the data to a centralized server, where clustering is performed. This has transmission of all the data to server results in large communication costs.

In [66],authors perform clustering at each node where data is generated, and merges these clusters into a single global clustering at a low cost. They estimate the cluster centers on the current set of data points on the local site. Every incoming data point is assigned to its closest center, while ensuring the distance is within a certain factor of an optimally computed radius. Otherwise, they calculate clustering again on current set of points. After applying the furthest point algorithm, the centers are transmitted to the central server, which then computes a global clustering from these local centers.

2.3 Synopsis techniques in data stream

The unbounded size of a data stream and constraints posed by limited resources makes it impossible to compute an exact query answer. We calculate an approximate query answer with respect to a summary of the data stream. This summary information is called synopsis of the data streams. We can design an approximate data structure that maintains a small synopsis of the data rather than an exact representation. Hence, keep computation per data element to a minimum. Various approaches have been proposed in literature for maintaining synopsis for query answering. These approaches are random sampling, sketches, histograms, wavelets, micro-clustering, density estimators etc.

Sampling methods are among the simplest methods for synopsis construction over data streams. We use them to provide information about the data by examining only a fraction of data. The random sampling involves choosing a subset of the stream, which reflects the main characteristics of the stream. The reservoir sampling can be applied to compute the sample over data stream. It scans the data in single pass over the data and provides an sample of the already processed elements. As reservoir sampling has been designed using temporal function to focus on recent data.

Histogram partitions data distribution along any attribute. The data distribution is divided into a set of ranges called buckets. These buckets have a count associated with them which maintains the count of data values present in that bucket. It does not maintain the distribution of the data values within a bucket and assumes the distribution within a bucket to be uniform. The space requirement depends on the number of buckets in the histogram. Histogram based methods [67]are widely applied on static data sets but their extension to the data stream problems is a challenging task.The V-optimal histogram approximate the distribution of a set of values by a piecewise constant function. This minimizes the sum of squared error. The equi-width histograms [68] partition the domain into buckets.The number of values falling into each bucket is uniform. They maintain

quantiles for the underlying data distribution as the bucket boundaries. They characterize data distributions in a manner that is less sensitive to outliers. They are useful in traditional databases for selectivity estimation.

End-Biased histograms maintain exact counts of items that occur with frequency above a threshold, and approximate the other counts by an uniform distribution. [69] showed how to compute optimal V-optimal histograms for a given data set using dynamic programming. [70] adapted this algorithm to sorted data streams and constructs an arbitrarily V-optimal histogram. [71] provide algorithms based on the sketching technique. They view each data element as an update to an underlying vector that we are trying to approximate using the best B-bucket histogram.

Sketches are random projections of a given data vector, which can be utilized to approximate the frequency moments of the data. It is an extension of the random projection technique in which we can reduce a data point of dimensionality d to an axis system of dimensionality k by picking k random vectors of dimensionality d. These k random vectors are selected from normal distribution and have zero mean and unit variance. These random vectors are normalized to one unit in magnitude. We calculate the dot product of the data values with each of these random vectors. The dot product of two vectors after transformation preserves the proportional Euclidean distances between the data points. Sketches have been used to estimate point and range queries over data streams. A function maps values of a finite domain to non-negative integers hence, maintaining a sketch of signal over data stream. We use sketches to estimate linear projections of the underlying signal. We can use sketches to recover most important wavelet coefficients. Sketch-based methods assume that the value of each stream element is from a finite domain. However in real-valued data streams have elements from infinite domain and need initial discretization. This induces errors in the result and adversely affects algorithmic complexity. Count-min sketches [72] support a variety of queries over a data stream and improve the time and space bounds in comparison to other approaches.

Wavelet transforms is a well known technique for hierarchical data decomposition and summarization [73]. It is a lossless function representation and decomposes the data into a sequence of wavelet coefficients. These coefficients are projections of the given data onto the orthogonal set of basis vector. Only a small number of these coefficients are sufficient to represent data characteristics. The majority of coefficient generated are insignificant and can be approximated to zero without loss of much information. Authors tend to minimize the reconstruction error by retaining only a fixed number of coefficients. The number of coefficients is defined by considering space constraints. The mean square error is optimized by selecting the proper range of coefficients for processing. Wavelets have been used in many applications in the field of image and query processing. Different types of wavelets have been proposed in literature. The Haar wavelet has been used widely due to its easier computations. We can create wavelet representation of the frequency distribution of a data along any dimension. The non normalized Haar coefficient provide the difference in relative frequencies in the equi-width histogram buckets. These Haar coefficient of different order corresponds to buckets of different levels of granularity. These different order coefficients provide an understanding of the trends in the data at a particular level of granularity. The low order wavelet coefficients give localized trend in data while higher order wavelet coefficients represent the broad trend in data. The Haar wavelets are easy to work with as they compute coefficients by averaging and differencing operations over data set.

In [74], authors propose a single-pass computation of a wavelet synopsis. The wavelet thresholding algorithm achieve excellent accuracy with respect to maximum-error metrics. They extend the technique to the streaming case. [75] introduces algorithm to create wavelet synopsis over data streams with respect to non-Euclidean error measures. Haar wavelets are used for multiple data streams are simple but the least smooth wavelet family hence, decreases the quality of wavelet representations.

Another way for computing approximate query answers is to consider only recent

elements using sliding window. Sliding window has been used in many applications. In some applications we require to approximate answer to a data stream query only on recent data from stream and not over the entire past history of the data streams. For example, queries to be performed over last month. That means we maintain data only for one month and keep on discarding old data. In this way we limit the data on which we want to perform queries. This method is quite useful in real–life applications where recent data is of more importance than old data. The method is easy to implement and works well with the evolving nature of data stream. For example movement of the stock in last week, visitors to the website in last week, phone call record. The queries performed on sliding window are normally continuous queries. They produce high quality approximation of query results due to deterministic nature of technique. However, when the data of sliding window is too large to be stored in main memory we need to design algorithms which work within the given memory. [16] maintain statistics using sketch based algorithm on sliding window model. [76] adapt the reservoir sampling algorithm to the sliding windows case and compute Iceberg queries over data streams. Manku and Motwani [77] present techniques to adapt their algorithms to the sliding window model. Guha and Koudas [78] provide a technique to maintain V-Optimal Histograms over the sliding window model.

Sliding windows are defined with respect to a timestamp or sequence number attribute representing arrival time of data value. In many applications such as sensor networks, comparing timestamps across stream elements may be relevant. When the system adds a special field to each incoming data value, it is called Implicit timestamps and when a data attribute is designated as the timestamp is called Explicit time stamp. Explicit timestamps are useful when each data value corresponds to an event at a particular time that adds meaning to the data. Implicit timestamps are useful when the exact moment in time associated with a data value is not important. It considers in general whether data value is old or recent. In explicit timestamps data value may not arrive in the same order as

their timestamps and makes it difficult to perform computations. This problem can be rectified by maintaining a buffer.

A probability distribution function gives the description of data distribution. It is useful in finding probability of a random variable. The density estimation is a construction of an estimate of density function from the observed data. Kernel density estimation is a statistical technique that approximates a probability distribution. In this technique we place a bump at each point and sum the height of these bumps to generate a uniform random sample. The kernel function defines the shape of the bump. It is a uni-modal symmetric, non-negative function that centers at zero and integrates to one. So the kernel density estimate is the overall sum of bumps centered at each observation X_i and bandwidth defines the width of each bump. [79] have used cubic splines to estimate continuous queries.

2.4 Selectivity estimation methods

The selectivity estimation involve storing the data in the memory for computation. The storage of data stream in memory is not possible due to large size of data streams. Hence, we maintain the summary of data in memory using various synopsis methods. Then we approximate selectivity over the summary of data. We survey various selectivity estimation techniques.

We can use sampling to represent the data stream by randomly selecting a fraction of data. We select a random sample [29, 80, 81, 82, 83, 84] from the large data stream and apply queries on this sample for selectivity estimation. The sampling representation is not specialized and uses the original multi-dimensional representation of data values in different applications. The introduction of the bias function improves the results of reservoir sampling over recent horizons of the data stream. This function is effective in dealing with evolving nature of data stream. The reservoir is restricted by the available memory space and cannot increase the sample size within the available memory restrictions. The concise samples

[83] overcome the space limitation problem of reservoir sampling. The technique maintains samples incrementally regardless of data distribution as ¡value,count¿ pairs. Counting samples keep the track of all the occurrences of a value selected for the sample. The value of count is increased for the sampled item. The technique deals with the size restriction problem by reducing the value of count for all the pairs in each pass and subsequently removes all the singletons from the sample. The concise sampling uses the fact that the number of distinct values of an attribute is significantly smaller than the size of the data stream. This method works better for one dimensional case, while the simple reservoir based method is more appropriate for multi-dimensional sampling.

Wu et al. [85] use Golden rule of sampling for effective range query estimation on non-uniform probability distribution function. They transform non-uniform function *f(x)* to uniform distribution by using cumulative distribution function *F(x)*. The random sample S over *F(x)* captures the essential characteristics of the distribution. Inverse of cdf function $F^{-1}(S)$ gives the values in the original domain *f(x)*. Selectivity is calculated as difference in cumulative distributive function(cdf) of end points of the range query. Wu et al.[86] use adaptive sampling on cdf for query estimation. The method uses range query result feedback to tune interpolation on cdf. An approach of estimating join queries is to use sampling to construct join synopsis [87, 88]. The technique pre-computes samples of join result without executing join queries. The technique obtain random samples of all joins using samples of results of small set of joins. The samples of small A newly arriving tuple T of relation R is added to the random sample S with the probability equal to size of S divided by size of R.

Sampling is easily adaptable to continuous data stream environment but it has quite a few drawbacks. A large sample is required to estimate accurately. It is not suitable for large data stream size as well as for large domain size. It does not work well with join queries and multi-dimensional data. Various histograms [89] have been proposed in the literature varying in the size of the range and frequency of data in a bucket. The equi-width histogram is a simple representation which

divides the data into equal length buckets and stores the frequencies of these buckets. These histograms compute buckets of fixed size and variable point frequency. The technique iteratively partitions the data space into multi-dimension buckets along each attribute. However equi-width partitions results in unequal distribution of data points across different buckets. Moreover, the buckets in the range boundary of a query may lead to inaccurate query estimations. The equi-depth buckets overcome the problem of unequal distribution by creating buckets with equal number of data values. The equi-depth histograms implementation requires the partitioning of ranges in advance which is not possible in case of data streams. Ioannidis et al. [90] have concluded that histograms can be chosen independently for each relation such that its self-join is optimal. The optimal histogram was proposed by [69] and was shown to be optimal for any other query applied on any other relation. The improved version of this histogram has been applied on data streams by [78].

We tend to minimize the frequency variance of the different values within a bucket, so as to ensure that the uniform distribution assumption is held for queries. It helps in minimizing the boundary error due to extrapolation in a query. [91, 92] introduced the V-optimal, maxdiff and compressed histograms to ascertain the selectivity of range predicates. These histograms differ in terms of source and sort of parameters. The V-optimal histograms minimize variance in frequency and area. This is quite useful in the multi-dimensional case, where the number of buckets can be very large due to combination of a large number of dimensions. The V-optimal histogram works well for selection queries. The skew in data is handled using compressed histograms. These histograms store highest frequencies in one bucket and remaining values in multiple buckets. The maxdiff histograms place boundaries between adjacent source values having large differences.

In case of multidimensional uncorrelated data, we create separate histograms for each attribute. We calculate selectivity of each attribute separately. Selectivity over multiple dimensions is evaluated by multiplying selectivity of each attribute. Min-Skew is a multidimensional histogram designed for spatial datasets [88]. It

approximates the original dataset using a regular grid. It computes the number of rectangles that fall inside each cell of the grid. Then partitions the array produced by the grid to create the buckets of the histogram covering the whole data space. The goal is to minimize the variance inside each bucket. Information in the buckets can be compressed using discrete cosine transformation [93]. We partition each dimension into uniform histogram buckets. The number of buckets for lower dimensions is less as compared to higher dimensions. Selectivity is estimated by integrating cosine coefficients over the query range.

The STholes method [94] estimates statistically independent data. It organizes multi-dimension histograms hierarchically as a tree and maintains parent child relation in-between buckets. Each node of the tree is a bucket which itself can be another histogram. The tuple count inside child bucket does not belong to parent bucket. The technique improves the estimate by considering query feedback while creating new buckets. It considers bucket merging to solve the limited memory size problem. STholes + [95] improves the STholes method by reducing the space required to store the bucket.

The 4-level tree index [96] based scheme approximates cumulative frequencies for each bucket in 32 bits. These bits are used to store the partial frequency sums at seven intervals inside the bucket and overall frequency sum of the bucket. The frequency values are organized in a 4-level tree over the bucket. The 4LT technique is applied to maxdiff and V-Optimal to improve range query estimation. A category of histograms do not partition the attribute domain for example wavelet based and binary tree based. These histograms are called non-bucket histograms. A wavelet based histogram contains a set of wavelet coefficients and information can be used for constructing original frequency. nLT is a non bucket based histogram used for range query estimation [97]. It is a hierarchal decomposition of the original data distribution kept in a binary tree, containing pre-computed hierarchal queries. The root node of this tree contains sum of all the frequencies of the domain. In a non-leaf node, the left child node contains the sum of frequencies of left hand half and the right child node contains the

sum of frequencies of right hand half. The domain is divided into 2^{n-1} equal size sub-ranges and the tree is constructed in bottom up fashion. The selectivity is estimated by visiting the tree nodes from root to leaf node. Variable size buckets [98, 99] have been used to solve multi-dimensional range queries over high correlated real attributes. The technique selects all the dense grids and makes each of them a bucket. These buckets are allowed to overlap. This ensures uniform data distribution within buckets, hence, improves the accuracy of approximation. Each bucket stores location identifier and the count of total number of tuples.

The data density of intersecting area of overlapping buckets is approximated as the sum of the data densities of overlapping buckets. For a tuple lying in the intersection of many buckets, we count only average density of one of the bucket. The algorithm iteratively approximates the density function using a grid and searches for buckets with the larger average density. The integral sum over all the buckets gives selectivity of an aggregate range query.

Histograms are concise and efficient for low dimensional data. In case of multi-dimensional datasets the number of buckets a query can intersect increases exponentially. The space requirement increases with increase in number of dimension. The large domains of attribute decreases efficiency, updates take lot of time.

Wavelets provide an easy way to approximate query answering [100, 101, 102]. Wavelets are used to hierarchically decompose a set of values into wavelet coefficients. We retain the largest absolute wavelet coefficients and provide an approximate summary of the original values. We can use a wavelet synopsis to reconstruct the approximate original values. After normalization the sum of the squares of these coefficients is equal to the energy in the data stream. These coefficients provide a new coordinate representation after axis rotation. In new representation the Euclidian distances are also preserved. Hence, we can approximately represent the series with a small number of coefficients. We minimize error by retaining only predefined number of coefficients from the decomposition. We use wavelets for efficient and approximate query processing of data streams.

They are quite useful for range queries as we can reconstruct the contiguous ranges with a small number of wavelet coefficients.

For applying wavelets on multi-dimension dataset, one-dimensional wavelet transform can be performed on the first dimension and the actual values are replaced with the resulting coefficients. Similarly for the second dimension, treating the this modified matrix as the original matrix, we continue up to d dimensions. We keep most weighted wavelet coefficients and reconstruct signal from these coefficients. [103, 104] use histogram along with haar wavelets to represent the data distribution. The technique selects a part of coefficients based on either taking largest coefficient or deleting coefficients. Selecting largest coefficients lead to large error reduction and deleting smaller coefficients lead to small error increase. Wavelet based histograms are a useful technique for selectivity estimation. Matias et al. [103] have used wavelets to compress bulky histograms. Histograms are built on distribution by using multi-resolution wavelet decomposition. The technique captures joint distribution of multiple correlated attributes. The preprocessing generates extended cumulative distribution from the data, and then wavelet decomposition gives a set of N Haar wavelet coefficients. The cumulative distribution at a given value b is constructed as the sum of coefficients on the error-tree. The technique retains m most significant coefficients for processing. The technique improves the accuracy of histogram, hence, improves selectivity estimation. It takes multiple passes for estimation. In the case of data streams, we would like to maintain the wavelet based histogram dynamically. Authors perform the maintenance with frequency distributions rather than cumulative distributions. Whenever a new data element arrives, the frequency distribution along a given dimension gets updated. Some of the wavelet coefficients may change and many of these coefficients may be small and may eliminated.

Chakrabarti et al. [105] performs general purpose query approximation using multi-dimensional Haar wavelets. They construct a compact wavelet-coefficient synopsis of relational table and evaluate various queries directly over wavelet synopsis. The select, project and join are executed entirely in the wavelet-coefficient

domain and the results are mapped from the wavelet domain to relational tuples. Wavelets have been used indirectly as compression technique over histogram and sketches. Data distribution is represented in space efficient manner but require large space to calculate wavelet coefficients. The updating of coefficients is quite complicated. A key issue for the accuracy of the selectivity estimation is the choice of coefficients to be retained. The selection of largest B normalized coefficients minimizes the mean square error criterion. However it is not always best to choose only the large coefficients. The more careful selection of coefficients can minimize specific error criteria i.e minimization of the mean square error or the maximum error metric. However the mean square error metric is a global optimization technique and ignores local behavior. This means query results performed on certain regions of the series such as recent time windows are poor. In case of data streams random vectors are defined by hash functions and stored in a matrix. We consider stream of data as vector and its dot product with the matrix of random vectors gives a sketch. This sketch is a linear transformation of data stream. Sketches store the updated information incrementally in small space i.e. sub linear in the data size. They can be updated quickly using small amount of memory. It is possible to maintain sketches in the presence of deletions.
Sketches have been used in various rotational invariant properties such as the L2 distance, dot product, or second moments are approximately preserved by the sketch. Sketches are quite useful in tracking frequencies of distinct values of a stream. In this case the sketch is defined as dot product of the stream vector with a random vector of N distinct values. The FM sketch [106] technique counts distinct items. It maintains a bitmap array of length $\log m$ and uses hash function to map various data items. Each item is mapped by a hash function into an entry of bitmap with certain probability. The information stored in sketches can be used to find approximate answers to queries.

Sketches have been used widely for selectivity estimation. The method of sketches can be effectively used for join estimation, which is calculated as inner product of frequency vectors. The AGMS sketch [107, 108] is a simple way of generating

summaries of data as a random variable. Whenever a data item e with frequency w arrives, the sketch vector is updated as $x_f[i] = x_f[i] + w * \xi_i(e)$, where $\xi_i(e)$ is a family of uniformly distributed, 4-wise independent random variables, with different independent families. The inner product of the frequency vectors $X[i] = x_f[i].x_g[i]$ is an unbiased estimator. The inner product can be used to estimate point and range queries by reducing it to the size of join computation. A group of independent sketches improves accuracy and calculates final estimate using averaging and selecting group median. We generate sketches $S_1 \ldots S_k$ for n data values, the expected value of S_i^2 is equal to the second moment provides an estimation for self-joins. If u_i be the number of items corresponding to the i^{th} value, then the second moment estimation is exactly the size of the self-join. The sketch partitioning [109] improves the method by partitioning the join attribute domains and then estimating the size of individual sub-domains. They maintain separate sketches of each partition and compute the join estimate as the sum over all partitions. The intelligent partitioning of problem require prior knowledge of data distribution. It improves the results by reducing variance of the estimate. The work was extended to multi-join queries based on join graph model of the input query [110]. The method shares sketch computations and space across multiple queries. Multiple queries are processed concurrently improving utilization of sketch space. The method requires prior knowledge of data distribution and independence assumption of join attributes. The skimmed sketch technique [111] improves the accuracy of result by reducing variance. It skims the dense frequencies from random hash-sketch summaries of two streams as separate distributions. They treat most frequent items responsible for high variance separately. The sub join sizes are estimated on these dense and sparse frequencies. The join size is calculated as sum of these sub join sizes. A skimmed sketch can be constructed by subtracting out the sketch components of these frequent items.

JD sketches [100] uses two level hash function to estimate Join distinct queries. The method relies on family of hash functions that map incoming elements uniformly and independently over collection of binary strings. It uses 2 level hash

sketches which are generalization of basic FM bitmap hash synopsis. Each bucket is an array of independent 2 level hash sketches built on the set of B values for the tuples. The two level hash sketches use one randomly chosen first level hash function applied to projected attributes. The second level hash function is applied on join attributes. Pairs of JD-sketches are built from the input streams which are used to generate bit map for distinct value pairs generated as intersection of first level buckets. The dot product of a data stream with the random vector may be used to compute the wavelet decomposition. Each wavelet coefficient may be computed as the dot product of the wavelet basis with the time series data stream, while their approximation may be computed by using their sketches. After computing coefficients we retain the B coefficients with the highest energy. We derive wavelet coefficients using the sketch representation of the frequency distribution of the original stream. The entire synopsis structure may need to be updated and every wavelet coefficient must be changed to find the best B coefficients.

The aggregate query answering on the signal has been estimated using wavelets and sketches [101, 112]. The technique first creates sketch of data items and then generates highest B wavelet coefficients. These coefficients are used to provide point wise and range sum estimation over data streams. Whenever a new item arrives the sketches are updated, subsequently B coefficients are updated. Sketch based methods takes space sub linear to data size in the presence of deletion. However they are not good for multi-dimension data space and difficult to apply to arbitrary applications. The difficulty of spontaneous analysis of sketch based representations, sometimes make them difficult to apply to every application. The application of sketch based methods to the multi-dimensional data sets is still an open problem.

The probability density function is unknown for an arbitrary data stream. Kernel density estimator over data streams [113, 114] is a given set of observations of an unknown random variable X. To determine the probability of X for interval $[a, b]$, we integrate density function with respect to $[a, b]$.

We use kernel density function for query estimation. In this techniques data is stored using smooth kernels. The overall number of these kernels is restricted due to the limited amount of available memory. We merge kernels to keep the number of kernel constant with minimum loss of accuracy. M-kernel merging [115] explore maintenance of linear kernel functions for density estimation. It uses normal function as kernel function and one kernel function to store more than one data points. These M-kernel functions have three parameters weight, mean and bandwidth. A weight value is given to each kernel function to indicate how many data points it will represent. The maximum number of Gaussian kernels is set for one dimensional data stream. A new M-kernel is inserted for each incoming element provided it is not equal to mean of an existing M-kernel. If it is equal to mean then weight of that kernel is incremented. Similar types of kernels are merged to process data efficiently in fixed memory together. The kernels having minimum merge cost are merged. A fadeout function is used to decrease the contribution of the old data in the data stream. M-kernels are over smoothed and fail to capture unknown density. Also the technique does not offer any theoretical foundation for the bandwidth. Heinz et al. [116] propose improvements in the M-kernel method. It has been noted that the shape of kernel function does not affect the approximation substantially. However the standard deviation of the function, or its bandwidth is quite important for accurate approximation. Therefore, we choose a kernel function that is easy to integrate. In this method authors use Epanechnikov kernel in place of Gaussian kernel. The Epanechnikov kernel is quite easy to integrate with minimum computations and deals with unbounded function. The magnitude of the bandwidth controls how far from the sample point the weight of the point is distributed. As the bandwidth becomes smaller, the non-zero diameter of the kernel becomes smaller. In addition to mean and weight used in M-kernel two variables minimum and maximum are added to kernel entries. This ensures that the complete data range will be covered. It uses global bandwidth for all the entries. The method introduces an intelligent merge scheme for kernel entries that adapt to changing system resources. The method

has linear processing cost with constant amount of allocated memory. There are two implementations one is list based and second is tree based. The list based implementation require lower processing cost and higher accuracy, while tree based implementation takes higher processing cost and lower accuracy.

The discrete wavelet transform divides a function into two components, where one component covers the general shape and the other component represents local features of data set. This helps in efficient compression of a function by removing the least important local details. The scaling functions create the general shape of function f, while the wavelets provide the local details. As f is not known, the sample $X_1, \ldots, X_n$ can be used for estimating the coefficients of the wavelet series expansion. The wavelet-based density estimates [117] performs well due to the local nature of wavelets. These estimators divide the data stream in to contiguous blocks and process the data block by block. They maintain a separate wavelet density estimator for each block. The linear combination of first m block estimators gives m^th over all estimators and successively merges them into one estimator. The technique updates the coefficients for each arriving element using all previous elements. Here authors use two functions, one performs a convex merging of the current overall estimator with a new block estimator. The second function compresses the estimator being returned by the merge step so as to fit it into the available memory. This reduces the convex linear combination of both expansions to the convex linear combination of the coefficients of their associated basis functions. The new overall estimator is again represented by its wavelet series expansion. The convex merge may result in number of coefficients which may exceeds the available memory. To overcome memory overflow problem the coefficient with least important local details of the wavelet series expansion are removed.

The main advantage of kernel density estimators is that the estimator can be computed very efficiently in one dataset pass, during which the dataset is sampled and the standard deviation along each attribute is computed. The density estimate at a given point, assuming far-off points have negligible contribution

to the sum. It is therefore we adapt the bandwidths of a kernel, centered at a specific point, according to the extension of the neighborhood of its center. The kernel will mainly contribute to the density estimation of points within that same local neighborhood. Kernel estimation method have two problems, first is setting of bandwidth parameters and the second problem dealing with large number of tuples in high dimensions. However, if the kernel density estimation is applied to very large data sets, it becomes computationally expensive and space intensive as it must get the data and keep them in the memory before doing estimation. A density estimate of function can be computed by using kernel-based density estimates. These estimates are accurate under certain assumptions.

Cosine series have excellent energy compaction property. They store most of the information about the data distribution in a few components of the transform. Yan et al. [118] generate a data density function using cosine coefficients. They maintain cosine coefficients dynamically for insert, delete and update. The integral over this function gives the range selectivity. The method is designed to work for both one dimension and multi-dimension datasets. Cosine series has been used for multi-join and equi-join size estimations [119].The technique normalize the data in the predetermined range i.e. [0,1]. It computes cosine coefficients for all the data points and filter high frequency coefficients using triangular sampling method. We maintain average of cosine coefficients for the dimensions to be joined and estimate the join size by adding up the products of the corresponding coefficients. Multiple joins are obtained by adding up the products of the corresponding coefficients on the same dimensions. Cosine transform has been used to approximate sum range queries over data cubes [120].

The goal of clustering is to partition a set of points into groups such that points within a group are similar in some sense and points in different groups are dissimilar in the some sense. The summarized data consists of the cluster centers together with the number of points assigned to that cluster. There exist many algorithms in the literature for generating clusters of a dataset. The summary

statistics of data stream clusters can be used for selectivity estimation. A multivariate parametric method for estimating the selectivity of window queries of large dimensional data sets has been proposed by [51]. SEC (selectivity estimation via clustering) uses EM clustering to extract as a mixture of Gaussian distributions. SEC represents each Gaussian by mean, variances, covariance and relative weight. K-clusters represented by Gaussian distribution are built randomly using EM clustering algorithm. The technique assign every data point to one of these clusters and updates mean, variance, covariance and weight of the cluster. These parameters are useful to find the probability with which a data point belongs to a cluster. Then algorithm calculates the combined probability for k clusters. The integral of interaction of query and cluster gives the selectivity of window queries. The result is further improved by decomposing query into rectangle queries and computing the integral of clusters. Then sum of the integral of decomposed window queries gives the final result. SEC+ improves SEC in term of result and space. It store only variance values in the covariance matrix. The points associated to one cluster assume attribute independence. In case of multi-dimension, Gaussian distribution is split into one dimension distributions. The integral is product of all one dimensional distributions. Micro-clustering method is useful in constructing synopsis over data streams. It can be applied to the multi-dimensional case, and can adjusts with the evolution of the underlying data stream. Aggarwal et al. [121] have used micro-clusters to estimate future queries for multidimensional data streams. They store snapshots of summary statistics at different point of time. The information stored in these snapshots is used to generate future statistical data for predicting query estimates. The techniques requiring clustering of dataset for query estimation can be accurate if the clusters themselves can be built accurately.

The entropy is measure of the uncertainty in random variable. The higher entropy means more unpredictability. It can be viewed as a way to capture information content, higher entropy means more information. It is estimated at several levels to measure uniformity of distribution. It can be computed incrementally and can

be used for selectivity estimation over continuous stream of data. We can find the entropy of a column and update it incrementally.

Similar to histogram, entropy maximization try to capture maximize information from the data. Maximum entropy is a technique for estimating probability distributions from data. The main idea in maximum entropy is that when nothing is known, the distribution should be as uniform as possible that is have maximal entropy. [122] have used maximum entropy principle that selects a probability distribution such as to maximize entropy function. The method chooses uniform selectivity model for selectivity estimation from already available selectivity. To et al. [123] present a technique to minimize reduction in total entropy of Histograms. They try to maximize the entropy by keeping the similar frequency data values in the same bucket. They propose algorithms to minimize the cumulative weighed variance of the bucket. Keeping similar frequency values in the same bucket is maximization of entropy of the column of frequencies. The incremental entropy-based histograms [124] for selectivity estimation are effective in accuracy and efficient in construction time. The bias factor and weighted mean as per bucket statistics improve estimation of equality query. Applying 4LT to the existing histograms such improves the accuracy of range query. The entropy of a histogram bucket is equal to the entropy of the probability distribution of the values within bucket. Probability of each value is defined as its frequency divided by total number of attribute values. For all buckets with the same number of attributes, bucket entropy is maximum when frequencies are equal.

In the next chapter we discuss the framework for range query estimation.

Chapter 3

Micro-cluster based framework for selectivity estimation

The micro-cluster based framework maintains clusters over data distribution. The advantage of the frame work is that it meets the processing requirements of data streams. The stream processing using micro-clusters is suitable to minimize number of operations per input, processing items in parallel and capturing data locality in one pass. In this chapter, we present micro-clustering and cosine series based framework for selectivity estimation over data streams. We start in Section 3.1 with a description of the data stream model. In Section 3.2 we discuss how we can maintain summary statistics of data stream using micro-clusters while section 3.3 describes how cosine series coefficients can be used as density estimators and used for selectivity estimation. Section 3.4 discusses the framework used for selectivity estimation of range queries. Section 3.5 and section 3.6 discuss the range query estimation over multiple data streams and distributed environment respectively.

3.1 Data stream model

A data stream consists of an unbounded sequence X_1,X_2, ... of d-dimensional, real valued numbers arriving at time stamps T_1, T_2, ...The data values are indexed with respect to the order in which they arrive. The data values can be inserted or deleted from the stream. In the data stream model, the input data values are not available for random access from memory, but arrive as one or more continuous data streams. These streams are considered as an ordered sequence of data. They must be accessed in order and can be read only once. We assume following about data stream model and algorithm for query estimation.

- Each arriving data value to be independent of previous data values.
- As found in most real world applications all the data values are independent of each other.
- The speed of arriving data items is quite fast.
- The data stream values evolve over the time. This results in change in the underlying data distribution.
- The data stream processing require, processing of each element only once within constant time.
- The query estimation requires data to be resident in the memory for finding accurate results.
- The algorithms need to maintain a tradeoff between the accuracy of the results and the time and space complexity.
- The amount of allocated memory is constant for unbounded and evolving data stream.

3.2 Maintaining summary of data using micro-clusters

We can represent the given data sequence in a compact form using clustering. The statistical information about the data locality can be maintained using micro-clusters. The micro-clusters are based on the cluster feature vector defined in [57], and store essential information to maintain the clusters and to estimate the query. We use the cluster statistics for selectivity estimation. The clustering process maintains both on-line and off-line components. The on-line component periodically stores summary statistics about the stream data is called micro-clustering.It does on-line processing and incrementally maintain the data summary. This component maintains the summary over whole data stream. We store the snapshots of these micro-clusters on the disk at different time granularity. This off-line component is called macro-cluster and used to answer various user queries over historical time horizons.

The micro-cluster MC_i stores the statistical information about all the data values of a cluster. This information can be used to maintain the cluster and calculate the data density estimation over the data values of that cluster. A *MC_i(N,S,SS,mCC,STS,SSTS)* contains the following information

- N:Number of data values in the cluster.
- S: Sum of all the data values in a cluster is used to calculate centroid, adding or deleting values from a cluster. The centroid or average of the cluster data values can be calculated as : S/N.
- SS: Square Sum of all the data values in a cluster is used to find standard deviation of data values.
  Radius of the cluster = $\sqrt{\frac{SS}{N} - (\frac{S}{N})^2}$
  Diameter of the cluster =$\sqrt{\frac{(2*N*SS)-(2*S)}{(N*N)-1}}$

- mCC: m number of Cosine series coefficients which are used to generate data density estimator for a particular cluster.This information is used for selectivity estimation.

- STS: Sum of Time stamps of data values is used to find the mean arrival time of data values in the cluster. It tells how old that cluster is, hence, make it possible to delete the micro-cluster with the least recent time-stamps.
  The average arrival time of data points in a cluster can be calculated as STS/N.

- SSTS: Sum of squares of time stamps of data items is stored to find the standard deviation. Approximate the average timestamp of the last data values of the cluster. When the least relevance stamp of any micro-cluster is below a user-defined threshold, it can be deleted.

These micro-clusters have additive property. We can easily add the statistics of the new arriving item, x at timestamp t to any cluster.

$$\begin{aligned} S &= S + x \\ SS &= SS + x^2 \\ N &= N + 1 \\ STS &= STS + t \\ SSTS &= SSTS + t^2 \end{aligned} \tag{3.1}$$

The cosine coefficients are updated by computing the average of the data with old coefficient values. Similarly we can merge two or more clusters by adding

their statistics.

$$\begin{aligned} S_C &= S_A + S_B \\ SS_C &= SS_A + SS_B \\ N_C &= N_A + N_B \\ STS_C &= STS_A + STS_B \\ SSTS_C &= SSTS_A + SSTS_B \end{aligned} \tag{3.2}$$

We update cosine coefficients by averaging with the coefficients of two merged clusters. We can subtract the statistics of one item from the micro-cluster. This is done by storing the micro-clusters at particular instance in the data stream as snapshots. We maintain the current snapshot of micro-clusters in the main memory. We can subtract the summary statistics of a cluster from another cluster to generate data in historical span. We subtract the micro-clusters stored at snapshots t_1 and t_2 to find statistics of the history for $t_2 - t_1$.

In order to store data in terms of micro-clusters, we assign every data value to a cluster. As fast and continuous data values arrive, they are assigned to nearest cluster. Nearness to a cluster is calculated as distance of data value from the centroid of that cluster. If the data value is not in the radius of the nearest cluster and total number of micro-cluster has not reached maximum, new cluster is created and the statistics is stored. We keep a maximum limit for the number of micro-clusters depending on the availability of the memory. If number of clusters has reached maximum then we delete cluster with the oldest time stamp. If there is no cluster old enough to be deleted then two nearest clusters are merged, followed by creation of new micro-cluster.

In the algorithm 1 we create clusters and maintain their statistics in a set of micro-clusters. Every new item is inserted into the nearest cluster. In case the data value does not fall in the radius of the nearest cluster we consider creation of

Algorithm 1 Algorithm to maintain list of micro-clusters

Input:
DS: $X_1, X_2, X_3, \ldots X_n$ stream of data values, each having two dimensions, normalized to (0,1)
Max : Maximum number of clusters
$S_i(S, SS, N, CC, STS, SSTS)$: Statistics of i_{th} micro-cluster
Output:
list: List having micro-clusters $MC_1, MC_2 \ldots MC_M$ in the increasing order of distance of centroid from the origin (0,0)

if list=NULL **then**
create a new micro-cluster MC_{new} and store the summary *S_i(S,SS,N,CC,STS,SSTS)*
else
Traverse the list to calculate distances of data point from centroid of micro-clusters i.e.$dist_1, dist_2, \ldots dist_m$
find $MC_{nearest}$ with $dist = MIN(dist_1, dist_2 \ldots dist_m)$
end if
if X_i falls in the radius of $MC_{nearest}$ **then**
allot X_i to $MC_{nearest}$
update the $S_{nearest}$
end if
if total clusters in the $list < Max$ **then**
create a new cluster MC_{new} and store the statistics *$Stat_i$(S,SS,N,CC,STS,SSTS)*
insert MC_{new} into the list
else
find the oldest micro-cluster MC_{oldest}in the list
delete MC_{oldest} from the list
create a cluster MC_{new} and store the statistics *$Stat_i$(S,SS,N,CC,STS,SSTS)*
insert MC_{new} into the list
end if
if there is no old cluster **then**
find two nearest clusters MC_i and MC_{i+1}
$MC_{merged} = MC_i + MC_{i+1}$
create a new cluster MC_{new} and store the statistics *$Stat_i$(S,SS,N,CC,STS,SSTS)*
insert MC_{new} into the list
end if

new cluster. We search the oldest cluster using the time stamps stored in micro-clusters and consider it deletion. Otherwise we consider merging two nearest cluster to create new cluster. The Statistics of two micro-clusters S_i and S_{i+1} to be merged can be added easily. S, SS, N, STS and $SSTS$ can be added directly while new average of coefficient is calculated for all the m coefficients. We can easily find whether the data value fall in the radius of the cluster using μ i.e. mean of all the data items σ i.e. standard distribution. These parameter can be calculated using S, sum of data values and SS square sum of data values stored as micro-cluster statistics.

These micro-clusters work with the large volume, and does not store the data explicitly on external disk. We process the data in one pass, in which all the summary information required for the clustering process needs to be stored and maintained within the small and constant time. It deals with continuously evolve stream over time by creating new clusters and removing obsolete clusters.

3.3 Cosine series to store summary of data values

We can describe the frequencies of attribute values by defining a data density function over the observed data. We use cosine series to construct data density function. The cosine series transforms the data using basis functions and has following properties

- De-correlation : Cosine series exhibits excellent de-correlation properties. It reduces the redundancy between neighboring data and leads to uncorrelated coefficients which can be encoded independently. Hence, it transforms the correlated data into uncorrelated coefficients. This de-correlation characteristic renders some reduction in the pre-computation complexity.

- Energy Compaction: Efficacy of a transformation method is measured by its ability to store input data into as few coefficients as possible. The most of the information is concentrated in a few low-frequency components of the transform. This allows the us to discard coefficients with relatively small information without lose of important information. Discarding some high frequency details can yield significant dividend in reducing the overall entropy. The cosine series has excellent energy compaction for highly correlated data. The energy compaction performance is optimal when data correlation approaches one.

$$f_i = \int_0^1 f(x)\phi_i(x)dx \qquad \text{for each } i \geq 0 \tag{3.3}$$

where sequence ϕ_i for r = 1,2,...

$$\begin{aligned} \phi_0(x) &= 1 \\ \phi_{2r-1}(x) &= \sqrt{2}\cos(r\pi x) \\ \phi_{2r}(x) &= \sqrt{2}\sin(r\pi x) \end{aligned} \tag{3.4}$$

and estimator of f_{i_1,i_2} for $i_1 = 0 \ldots m-1$ and $i_2 = 0 \ldots m-1$ for $X_1, X_2 \ldots, X_n$

$$\hat{f}_{i_1,i_2} = \frac{1}{n}\sum_{j=1}^{n}\prod_{k=1}^{2}\phi_{i_1,i_2}(x_{j,k}). \tag{3.5}$$

The cosine series consist of infinite functions with $1, \sqrt{2}\cos(1\pi x), \sqrt{2}\cos(2\pi x)$, $\sqrt{2}\cos(3\pi x), \ldots$

$$f(x) = a_0 + a_1\cos(x) + a_2\cos(2x) + \ldots f(x) = a_0 + \sum_{n=1}^{\infty} a_n\cos(nx) \tag{3.6}$$

These functions work as orthonormal basis for distribution selectivity. If n is the length of input sequence, coefficients of data density estimator can be calculated

and updated easily for X_j.

$$\hat{\beta}_0 = 1.$$

$$\hat{\beta}_1 = \frac{1}{n}\sum_{i=1}^{n}\sqrt{2}cos(\pi i).$$

$$\hat{\beta}_2 = \frac{1}{n}\sum_{i=1}^{n}\sqrt{2}cos(2\pi i).$$

$$\hat{\beta}_3 = \frac{1}{n}\sum_{i=1}^{n}\sqrt{2}cos(3\pi i).$$

$$\ldots \tag{3.7}$$

$$\ldots \tag{3.8}$$

$$\hat{\beta}_{(m-1))} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{2}cos((m-1)\pi i).$$

Estimator of data density function can be given as:

$$\hat{f}(X) \;=\; 1 + \hat{\beta}_1\sqrt{2}\cos(\pi i) + \hat{\beta}_2\sqrt{2}\cos(2\pi i) + \ldots \hat{\beta}_{(m-1)}\sqrt{2}\cos((m-1)\pi i). \tag{3.9}$$

Integration of this function gives the estimation

$$\hat{\sigma}_{sel} = \int_a^b \hat{f}(x)dx. \tag{3.10}$$

On insertion of an element X, $\hat{\beta}_i$ is updated by calculating average coefficients for *n+1* data values.

$$\hat{\beta}_i = \frac{\hat{\beta}_i * n + \sqrt{2}\cos(\pi i x)}{n+1}. \tag{3.11}$$

On deletion of an element X, $\hat{\beta}_i$ is updated by calculating average coefficients for *n-1* data values.

$$\hat{\beta}_i = \frac{\hat{\beta}_i * n - \sqrt{2}\cos(\pi i x)}{n-1}. \tag{3.12}$$

3.4 Range query estimation using Data Density Functions

Range queries are used to find number of data values falling in a particular range. These are normally of the form $a \leq x \leq b$, here x is an attribute in the range a to b, whereas a and b are constant values. When a and b are equal it gives estimation at a point. e.g. Find the number of persons having height greater than 160 cm and less than 170 cm. Selectivity of range queries tells us how much percentage of total values satisfies a predicate. Data density functions can be used for estimating the selectivity of range queries. Consider random quantity x that has probability density function, and then f gives natural description of x, using which probabilities associated with x can be found. Density estimation is construction of an estimate of density function from the observed data and used to estimate how many data values lie in a particular range.

Selectivity is estimated using the information stored in the micro-clusters. To estimate range query the micro-clusters within query range are selected and average of cosine coefficients are calculated. These coefficients are used for generating data distribution function, which is used to calculate the number of values lying in the range query.

In the algorithm 2 we find estimate the selectivity of range query for single dimension dataset. We use the statistics stored in micro-clusters generated using algorithm 1. We search for the micro-clusters which fall in the range of the query and use cosine coefficients stored in these clusters to estimate selectivity. In first step we search the linked list of micro-clusters to find the micro-cluster nearest to left value of the range a i.e.MC_a. We store the m cosine coefficients in the

Algorithm 2 Algorithm to estimate selectivity of range queries

Input:
list: List of micro-clusters in the increasing order of distance from the origin.
a,b: range of the query
Output:
result : Number of data values in the given range

Traverse the list and find cluster nearest to 'a'
$MC_a = FindNearestCluster(list, a)$
$storeCosineCoefficient(a, coa[1], coa[2] \ldots coa[m])$
$storeCosineCoefficient(b, cob[1], cob[2] \ldots cob[m])$
$n1 = n2 = N$
if $MC_a \neq$ nearest to 'b' **then**
  while $MC_i \neq MC_b$ **do**
    $updateCoefficients(cob[1 \ldots m])$
    $n2 = n2 + N$
    jump to next cluster in the list
  end while
  for $i = 1$ to m **do**
    $fa = fa + \frac{coa[i] * \sqrt{2} \sin(\pi i a)}{\pi i}$
    $fb = fb + \frac{cob[i] * \sqrt{2} \sin(\pi i b)}{\pi i}$
  end for
end if
if $fa < 0$ **then**
  $fa = 0$
end if
if $fb < 0$ **then**
  $fb = fa = 0$
end if
if $fa > 1$ **then**
  $fa = 1$
end if
if $fb > 1$ **then**
  $fb = 1$
end if
$result = fb * n2 – fa * n1$
return(result)

$coa[1 \ldots m]$ and $cob[1 \ldots m]$ where $coa[1 \ldots m]$ represent the cosine coefficient in micro-cluster MC_a and $cob[1 \ldots m]$ is the average of cosine coefficients of micro-clusters from MC_a to MC_b. Now find micro-cluster MC_b nearest to right value of the range i.e. b. The $coa[1 \ldots m]$ represent the cosine coefficient in micro-cluster MC_a and $cob[1 \ldots m]$ is the average of cosine coefficients of micro-clusters from MC_a to MC_b. In case MC_a is not same as MC_b then traverse the list up to micro-cluster MC_b nearest to b is found and keep averaging $cob[1 \ldots m]$. *fa* gives selectivity of a in MC_a and *fb* gives selectivity of b from MC_a to MC_b. *n1* gives number of data items in MC_a and *n2* gives number of data items from MC_a to MC_b.

3.4.1 Selectivity over historical data

In some applications we require estimation of queries over the historical data. The selectivity over historical data streams is a challenging task as we cannot store the large data streams. This problem is solved by storing the summary statistics of the micro-clusters at different snapshots of time granularity. This statistics is stored on the disk and we use it to compute the statistics for a particular span of time. This is computed by subtracting the information stored at two different snapshots. For example $MC_i(t1)$ and $MC_i(t2)$ are the summary statistics stored at times *t1* and *t2* respectively for MC_i where $t1 \geq t2$. The difference of two statistics gives the summary of data arrived from time *t1* to *t2* for the cluster

MC_i.

$$
\begin{aligned}
S_{t1-t2} &= S_{t1} - S_{t2} \\
SS_{t1-t2} &= SS_{t1} - SS_{t2} \\
N_{t1-t2} &= N_{t1} - N_{t2} \\
STS_{t1-t2} &= STS_{t1} - STS_{t2} \\
SSTS_{t1-t2} &= SSTS_{t1} - SSTS_{t2} \\
cc[j]_{t1-t2} &= \frac{cc[j]_{t1} * N_{t1} - cc[j]_{t2} * N_{t2}}{N_{t1} - N_{t2}}
\end{aligned} \tag{3.13}
$$

All the cosine coefficients are updated and we generate summary statistics for the time duration $t1 - t2$. We estimate the range query over the duration $t1 - t2$ by integrating data density function.

3.5 Selectivity estimation over multiple data streams

Let us assume that there exists N number of sources generating stream of data. Selectivity of range queries over these multiple data streams can be useful in analyzing the cumulative trends in data generated by different sources.

Let us assume that the data stream consists of a sequence of data values $X_1 \ldots X_k$, arriving at various time stamps. Then a micro-cluster stores the information about the all the data values of a cluster. This information is sufficient to maintain the cluster and to find the data density estimation over the multiple streams. A micro-cluster contains the following information

- S: Sum of all the data values in a cluster is useful in calculating centroid of the cluster
- SS: Square Sum of all the data values in a cluster is useful to find standard deviation of data values.

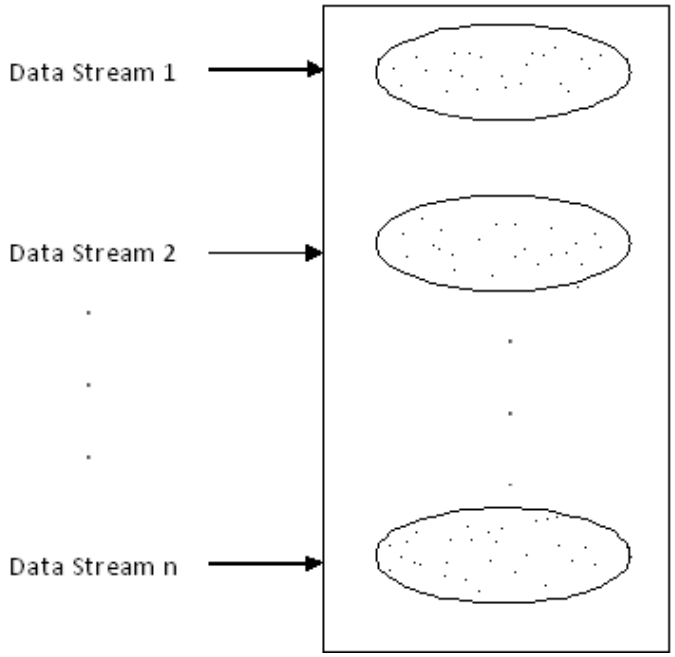

FIGURE 3.1: Multiple data streams

- N:Number of data values in the cluster
- mCC: m number of Cosine coefficients for generating data density estimator for a particular cluster.

As the data value arrives, we measure its distance from the centroid of different clusters. The data value is assigned to the nearest micro-cluster. If the data value is not in the radius of nearest cluster, a new cluster is created and the statistics is stored. We consider merging of clusters when they fall within the cluster radius. Algorithm 3 maintains micro-clusters over a data stream and is applied to all the data streams.

Any two micro-clusters can be merged easily by adding their summary statistics. *Sum, SquareSum* and N can be added directly while new average of coefficient is calculated for all the m cosine series coefficients.

We estimate the selectivity using the information stored in the micro-clusters. We compute the selectivity over multiple streams. Firstly we select the set of streams on which we want to apply the range query. In the next step we search

Algorithm 3 Maintaining micro-clusters over multiple stream

Input:
DS: stream of data values normalized to (0,1).
Si(S,SS,N,CC): Statistics of i_th micro-cluster
Output:
list: List of micro-clusters $MC_1, MC_2 \dots MC_m$

if list is empty **then**
  Create micro-clusters using K-means algorithm and add statistics to them
else
  Traverse the list to find the micro-clusters nearest to x
  $MC_nearestwith = MIN(dist_{1x}, dist_{2x} \dots dist_{mx})$
end if
if point falls in the radius **then**
  $AllotXtonearestcluster MC_nearest$
else
  create new cluster and store summary statistics
end if
Find the two clusters fall in the range
$MC_{merged} = MC_n + MC_m$
return(list)

all the micro-clusters of selected stream which fall in the range of the query. These selected clusters are used to generate aggregate statistics as a new micro-cluster. This new micro-cluster having aggregated statistics is used for range query estimation.

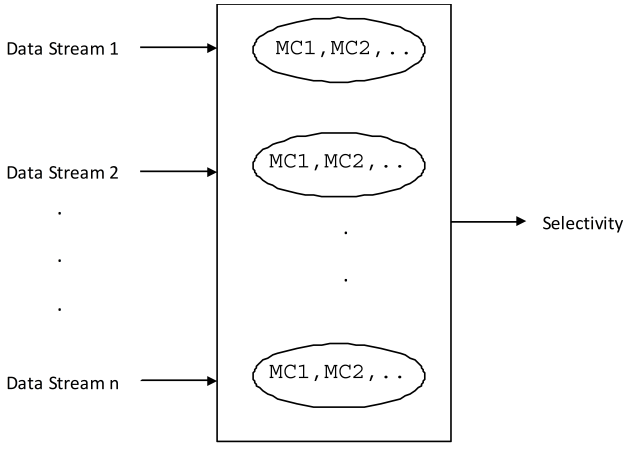

FIGURE 3.2: Selectivity over multiple data streams

We calculate the average of cosine coefficients over all the aggregated clusters. These coefficients are used for generating data distribution function, which is

used to calculate the number of values lying in the range query.

Algorithm 4 Maintaining micro-clusters over multiple stream

Input:
$list_1, list_2, \ldots list_N$: List of micro-clusters for N data streams.
a,b: range of the query
Output:
result : Number of data values in the given range

Select the set of streams for selectivity estimation from the $list_1, list_2 \ldots list_N$
for all the selected lists **do**
  Traverse the list and find clusters in the range a to b
  Add the summary statistics of all the selected clusters to MC_{new}
end for
for all the cosine coefficients in MC_{new} **do**
  $result = result + \frac{cc[i]*\sin(\pi i a)}{\pi i}$
end for
if $result \leq 0$ **then**
  $result = 0$
end if
return(result)

In the algorithm 4 we select a set of streams for the selectivity estimation. We use the lists of micro-clusters stored using algorithm 3. All the selected lists are processed to find the micro-clusters in the range of query. We add the summary statistics of all the micro-clusters in a new micro-cluster MC_{new} which fall in the query range. We use cosine coefficient stored in MC_{new} to estimate the selectivity of range query $a \leq X \leq b$.

3.6 Selectivity estimation over distributed data streams

In several real-life applications, such as wireless network analysis, data are naturally distributed among several sites. In many cases data are produced locally in large volumes and immediately moved to the centralized locations for analysis. In case of continuous and unbounded data streams we can not send all the received data to the central terminal as the cost of communication is quite large.

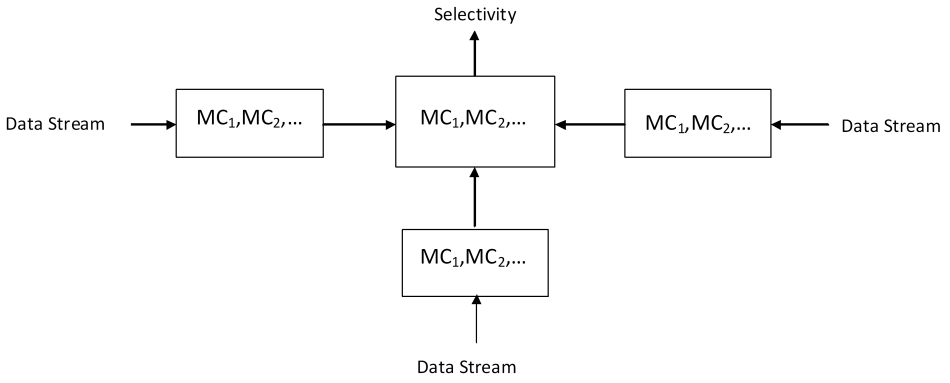

FIGURE 3.3: Selectivity over distributed data streams

We can maintain the data at various sites using our micro-clusters and communicate this summary information to the central site. The centralized location receives the summary information about the clusters from different sites and process this information for further analysis. The micro-clusters received from different sites are stored at different time granularity as snapshots. We generate aggregated information by merging micro-clusters. We estimate range query on the data using aggregated information stored in cosine coefficients of different clusters. The query estimation over a site can be performed by applying the range query over the stored snapshots.

The next chapter discusses the selectivity estimation of rectangular queries using micro-clustering.

Chapter 4

Query estimation of rectangular queries

With the increase in dimension, the task of selectivity estimate becomes difficult. We require large memory space and more computation cost for efficient and accurate estimation. Selectivity estimation of rectangular query is the percentage of total values of a data stream satisfies a particular two dimensional range. In this chapter, we present micro-clustering and cosine series based framework for selectivity estimation of rectangular queries over data streams. We start in Section 3.1 describing the rectangular query estimation problem. In section 3.2 we discuss the data density function for rectangular queries. In Section 3.3 we discuss how we can maintain summary statistics two dimensional data stream using micro-clusters while section 3.4 describes the framework used for selectivity estimation of rectangular queries.

4.1 Rectangular queries

Selectivity estimation of rectangular query is the percentage of total values of a data stream satisfies a particular two dimensional range. Let us assume that the

data stream consists of a sequence of data values $X_1 \ldots X_k$, arriving at timestamps $T_1 \ldots T_k$. Let us assume every data value consist of two attributes i.e. x and y. Rectangular queries are of the form $a \leq x \leq a_1 \wedge b \leq y \leq b_1$, here a, b, a_1 and b_1 are constant values, whereas x and y are attributes in the ranges a to a_1 and b to b_1 respectively. Figure 4.1 shows the examples of rectangular queries. We estimate the selectivity of rectangular queries for evolving data streams using

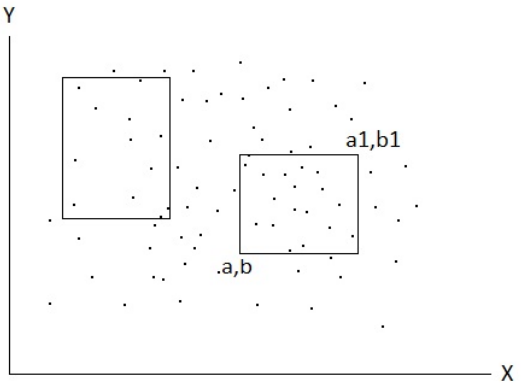

FIGURE 4.1: Example of rectangular queries

micro-clustering and cosine series. The method maintains micro-clusters having summary information about the data and its density distribution using cosine series. The method deals with evolving nature of data stream. The main features of the technique are mentioned as follows:

- Formulation of the selectivity problem in a way that is appropriate for rectangular queries over data stream.
- Online data stream processing algorithm that scans the data only once.
- The cosine estimator and micro-clusters estimates the selectivity while dealing with evolving nature of data streams. We can extend the technique to find selectivity over historical data by storing micro-clusters periodically off-line.
- The technique is local in which selectivity is estimated only by considering relevant clusters.

- The technique incrementally maintains the summary statistics. It reflects dynamic data updates to the statistics immediately.

4.2 Data density estimation of rectangular queries

We can use data density functions for estimating the selectivity of rectangular queries. Consider random quantity x that has probability density function f, then f gives natural description of x, using which probabilities associated with x can be found. Density estimation is construction of an estimate of density function from the observed data.

$$\hat{\sigma}_{sel} = P(a \leq x \leq a_1, b \leq y \leq b_1) = \int_a^{a_1} \int_b^{b_1} f(d)dydx. \tag{4.1}$$

The above probability density function can be used to estimate the how many data values exist in a particular rectangular range.

4.2.1 Cosine series as orthonormal basis

Orthonormal series estimators estimate the density for the unit interval [0,1] by estimating the coefficient of the expansion. For example f can be represented as the Fourier series $\sum_{i=0}^{\infty} f_i\phi_i$

$$f_i = \int_0^1 f(x)\phi_i(x)dx \qquad \text{for each } i \geq 0 \tag{4.2}$$

where sequence ϕ_i for $r = 1,2,\ldots$

$$\begin{aligned} \phi_0(x) &= 1 \\ \phi_{2r-1}(x) &= \sqrt{2}\cos(r\pi x) \\ \phi_{2r}(x) &= \sqrt{2}\sin(r\pi x) \end{aligned} \tag{4.3}$$

and estimator of f_{i_1,i_2} for $i_1 = 0 \ldots m-1$ and $i_2 = 0 \ldots m-1$ for $X_1, X_2 \ldots, X_n$

$$\hat{f}_{i_1,i_2} = \frac{1}{n} \sum_{j=1}^{n} \prod_{k=1}^{2} \phi_{i_1,i_2}(x_{j,k}). \tag{4.4}$$

Density estimate is given by

$$\hat{f}(X) = \sum_{i=0}^{k} \hat{f}_{i_1,i_2} \phi_{i_1,i_2}(X). \tag{4.5}$$

Cosine series has very good compaction property, as most of the signal information is concentrated in low frequency components. It consist of infinite functions that work as orthonormal basis for distribution selectivity. If n is the length of input sequence, coefficients of data density estimator can be calculated and updated easily for X_j.

$$\begin{aligned}
\hat{\beta}_{00} &= \frac{1}{n} \sum_{j=1}^{n} \sqrt{2}cos(0\pi x_{j,1}) \sqrt{2}cos(0\pi x_{j,2}). \\
\hat{\beta}_{01} &= \frac{1}{n} \sum_{j=1}^{n} \sqrt{2}cos(0\pi x_{j,1}) \sqrt{2}cos(1\pi x_{j,2}). \\
\hat{\beta}_{10} &= \frac{1}{n} \sum_{j=1}^{n} \sqrt{2}cos(1\pi x_{j,1}) \sqrt{2}cos(0\pi x_{j,2}). \\
\hat{\beta}_{11} &= \frac{1}{n} \sum_{j=1}^{n} \sqrt{2}cos(1\pi x_{j,1}) \sqrt{2}cos(1\pi x_{j,2}). \\
\hat{\beta}_{12} &= \frac{1}{n} \sum_{j=1}^{n} \sqrt{2}cos(1\pi x_{j,1}) \sqrt{2}cos(2\pi x_{j,2}). \\
&\ldots \\
&\ldots \\
\hat{\beta}_{(m-1)(m-1)} &= \frac{1}{n} \sum_{j=1}^{n} \sqrt{2}cos((m-1)\pi x_{j,1}) \\
&\qquad \sqrt{2}cos((m-1)\pi x_{j,2}).
\end{aligned} \tag{4.6}$$

Estimator of data density function can be given as:

$$\begin{aligned}\hat{f}(X) = \hat{\beta}_{00}\sqrt{2}\cos(0\pi x_{j,1}) * \sqrt{2}\cos(0\pi x_{j,2}) \\ + \hat{\beta}_{01}\sqrt{2}\cos(0\pi x_{j,1}) * \sqrt{2}\cos(1\pi x_{j,2}) \\ + \hat{\beta}_{10}\sqrt{2}\cos(1\pi x_{j,1}) * \quad \sqrt{2}\cos(0\pi x_{j,2}) + \ldots \\ \hat{\beta}_{(m-1)(m-1)}\sqrt{2}\cos((m-1)\pi x_{j,1}) \\ \sqrt{2}\cos((m-1)\pi x_{j,2}). \quad (4.7)\end{aligned}$$

Integration of this function gives the estimation

$$\hat{\sigma}_{sel} = \int_a^{a_1}\int_b^{b_1} \hat{f}(x,y)dydx. \quad (4.8)$$

On insertion of an element $X, \hat{\beta}_{i_1 i_2}$ is updated by calculating average coefficients for *n+1* data values.

$$\hat{\beta}_{i_1,i_2} = \frac{\hat{\beta}_{i_1,i_2} * n + \sqrt{2}\cos(\pi i_1 x_{j,1}) * \sqrt{2}\cos(\pi i_2 x_{j,2})}{n+1}. \quad (4.9)$$

On deletion of an element $X, \hat{\beta}_{i_1 i_2}$ is updated by calculating average coefficients for *n-1* data values.

$$\hat{\beta}_{i_1,i_2} = \frac{\hat{\beta}_{i_1,i_2} * n - \sqrt{2}\cos(\pi i_1 x_{j,1}) * \sqrt{2}\cos(\pi i_2 x_{j,2})}{n-1}. \quad (4.10)$$

4.3 Maintaining micro-clusters for selectivity estimation

In the proposed method, we maintain cluster over data stream and the arriving data values are assigned to the nearest cluster. We store the statistical information of all the data values in terms of micro-cluster, which is based on cluster

feature vector (Zhang et al., 1996). We use this information stored in micro-clusters to ascertain the selectivity of query. Let us assume $X_1 \ldots X_k$, are data values arriving at time stamps $T_1 \ldots T_k$. A micro-cluster MC_i stores the statistical information about all the data values of a cluster. This information is useful to maintain the cluster and to calculate the data density estimation over the data values of that cluster. A MC_i contains the following information:

- S_x, S_y : Sum of all the data values of attributes x and y in the cluster. These values are used to calculate centroid of the cluster, standard deviation etc.
- SS_x, SS_y: SquareSum of all the data values of attributes x and y in a cluster to find standard deviation of data values.
- N:Number of data values in the cluster
- CC: Cosine coefficients to generate data density estimator for a particular cluster.
- STS: Sum of Time stamps of data items to find the mean arrival time of data values in the cluster. It tells how old that cluster is; hence, make it possible to delete the micro-cluster with the least recent time-stamps.
- SSTS: Sum of squares of time stamps of data items; approximate the average timestamp of the last m data values of the cluster. When the least relevance stamp of any micro-cluster is below a user-defined threshold, it is removed and a new micro-cluster can be created.

As fast and continuous data values arrive, we assign them to the nearest micro-cluster. We find the nearest cluster by calculating the ecludian distances of the data value from the centroid of clusters. If the data value does not fall in the radius of existing clusters and total number of micro-clusters is not maximum, we create a new cluster. If number of clusters reaches maximum number then we delete the cluster with oldest time stamp. We merge two nearest clusters if there

is no cluster old enough to be deleted, followed by creation of new micro-cluster. The algorithm1 describes the maintenance of micro-clusters as synopsis.

Algorithm 5 Algorithm to maintain list of micro-clusters

Input: DS: $X_1, X_2, X_3, \ldots X_n$ stream of data values, each having two dimensions, normalized to (0,1)
Max : Maximum number of clusters
$Stat_i(S_x, S_y, SS_x, SS_y, N, CC, STS, SSTS)$: Statistics of i_{th} micro-cluster
Output: list: List having micro-clusters $MC_1, MC_2 \ldots MC_M$

if list=NULL **then**
  create a new micro-cluster MC_{new} and store the summary $Stat_i(S_x, S_y, SS_x, SS_y, N, CC, STS, SSTS)$
else
  Traverse the list to calculate distances of data point from centroid of micro-clusters i.e. $dist_1, dist_2, \ldots dist_m$
  find $MC_{nearest}$ with $dist = MIN(dist_1, dist_2 \ldots dist_m)$
end if
if X_i falls in the radius of $MC_{nearest}$ **then**
  allot X_i to $MC_{nearest}$
  update the $S_{nearest}$
end if
if total clusters in the list ¡ Max **then**
  create a new cluster MC_{new} and store the statistics $Stat_i(S_x, S_y, SS_x, SS_y, N, CC, STS, SSTS)$
  insert MC_{new} into the list
else
  find the oldest micro-cluster MC_{oldest} in the list
  delete MC_{oldest} from the list
  create a cluster MC_{new} and store the statistics $Stat_i(S_x, S_y, SS_x, SS_y, N, CC, STS, SSTS)$
  insert MC_{new} into the list
end if
if there is no old cluster **then**
  find two nearest clusters MC_i and MC_{i+1}
  $MC_{merged} = MC_i + MC_{i+1}$
  create a new cluster MC_{new} and store the statistics $Stat_i(S_x, S_y, SS_x, SS_y, N, CC, STS, SSTS)$
  insert MC_{new} into the list
end if

We select MC_{oldest} from the list based on the time stamps stored in micro-clusters. When we merge two micro-clusters,we add their statistics i.e S_i and S_{i+1}. We add $Sum, SquareSum, N, STS$ and $SSTS$ directly while calculating new average of cosine coefficient for the merged micro-cluster.

4.4 Selectivity estimation using micro-clusters

We estimate selectivity of rectangular queries using the information stored in micro-clusters. We generate data distribution function using cosine coefficients, which is used in calculating the number of values lying in the particular range query.

Algorithm 6 Algorithm for selectivity estimation(2dRQE)

Input:
list: List of micro-clusters MC_1, $MC_2 \ldots MC_{Max}$.
(a,b),(a_1,b_1): the range of query $a \leq a_1$ and $b \leq b_1$
m^2: Total number of coefficients
Output:
result : Number of data values from (a,b) to (a_1,b_1)

Traverse the list of micro-clusters and find the micro-clusters which fall in the query range. Store and update coefficients *coa* by averaging
for $i \leftarrow 0$ to $m-1$ **do**
  for $j \leftarrow 0$ to $m-1$ **do**
    if $(i+j) \leq m-1$ **then**
      if $i \neq 0$ and $j \neq 0$ **then**
        $K = (\sqrt{2}\frac{\sin(i\pi a_1)}{i\pi} - \sqrt{2}\frac{\sin(i\pi a)}{i\pi}) * (\sqrt{2}\frac{\sin(j\pi b_1)}{j\pi} - \sqrt{2}\frac{\sin(j\pi b)}{j\pi})$
      else if $i = 0$ and $j = 0$ **then**
        $K = (a_1 - a) * (b_1 - b)$
      else if $i = 0$ **then**
        $K = (a_1 - a) * (\sqrt{2}\frac{\sin(j\pi b_1)}{j\pi} - \sqrt{2}\frac{\sin(j\pi b)}{j\pi})$
      else if $j = 0$ **then**
        $K = (\sqrt{2}\frac{\sin(i\pi a_1)}{i\pi} - \sqrt{2}\frac{\sin(i\pi a)}{i\pi}) * (b_1 - b)$
      end if
      $sel = sel + coa[i][j] * K$
    end if
  end for
end for
if $sel < 0$ **then**
  $sel = 0$
end if
if $sel > 1$ **then**
  $sel = 1$
end if
return sel

The algorithm 6 describes the method of rectangular query estimation. We search to find all the micro-clusters lying in the query range. While finding micro-clusters we store the average cosine coefficients as *coa*.This gives the average of

cosine coefficient for query range. Selectivity,*sel* for of range query from (a,b) to (a_1,b_1)is estimated over *coa*. Algorithm 6 returns zero if *sel* is less than zero and one if *sel* is greater than 1.

4.4.1 Triangular sampling method

In multi-dimensional data streams the number of coefficients increases exponentially with the increase in dimensionality. The cosine series has very good compaction property and it stores the major information in small number of coefficients. We use this compaction property by considering only small number of coefficients for query estimation. This significantly decreases the cost of computation at the cost of accuracy. The selection of these cosine coefficients are sampled using triangular sampling which is used in digital signal processing. The triangular sampling method selects low frequency coefficients and work as a low frequency filter. This means only transformed coefficients within a specified zone are processed while remaining are set to zero. This selection of coefficients corresponds to low frequency filtering. In a 2-dimensional case the triangular method select the $(i, j)^{th}$ coefficients which lie within the triangle i.e. the sum of i and j is less then m. The number of coefficients selected by the triangular zonal sampling are calculated as by ${}^{n+b}\mathbf{C}_{min(n,b)}$, where n is number of dimensions.

TABLE 4.1: The number of DCT coefficients selected by the triangular zonal sampling for different values of n and b

	b=1	**b=2**	**b=3**	**b=4**	**b=5**	**b=6**
n=1	2	3	4	5	6	7
n=2	3	6	10	15	21	28
n=3	4	10	20	35	56	84
n=4	5	15	35	70	126	1210
n=5	6	21	56	126	1252	1462

As the dimensionality of data set increases, the number of coefficients chosen by the triangular zonal sampling increases slowly. The distribution should have high correlation among data items. This means the frequency spectrum of the

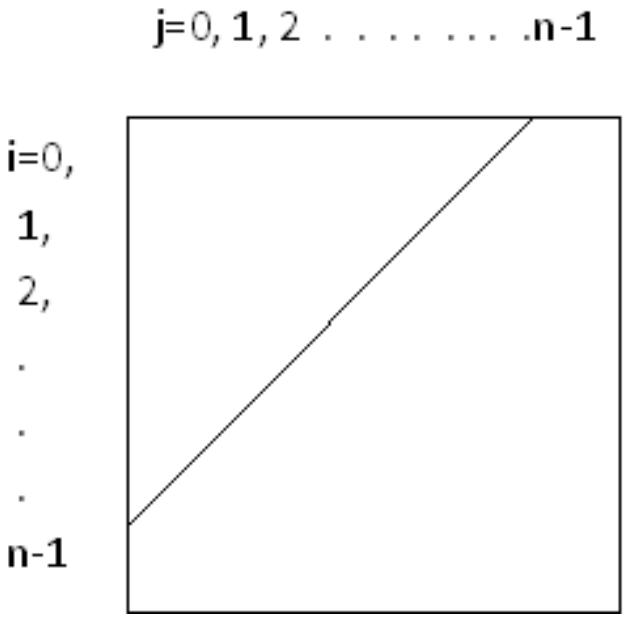

FIGURE 4.2: Triangular sampling

distribution should show large values in its low frequency coefficients and small values in its high frequency coefficients. In case the data are totally independent of adjacent data, we cannot have the benefits of energy compaction and we cannot reduce the number of coefficients without distorting the original data distribution. The real-life data distributions are highly correlated. It is evident for the joint data distribution of multiple attributes to have clusters in most cases, as the attributes are closely dependent on each other. As the skewness of data distributions grow or the number of clusters increases, the number of large-valued high frequency coefficients tends to increase.

In this chapter an approach for selectivity estimation of rectangular queries over data stream has been discussed. To solve the problem, we used micro-clusters as data stream synopsis. The summary statistics of data values were stored in micro-clusters in using cosine coefficients. The method calculated the selectivity using the integral of cosine functions over micro-clusters. The method processes the data stream in one pass and inclemently updates the summary statistics. It dealt with evolving nature of data stream and can be used for estimation on

queries on historical data. The results of the techniques has been verified for different sizes and types of datasets and are discussed in the next chapter.

Chapter 5

Experimental Evaluation

The previous chapters thoroughly discussed techniques we developed for estimating queries over data stream. We compared our techniques as well as competitive ones in an extensive experimental study, whose results we discuss in this chapter. In section 5.1 we discuss the results of experiments for single dimension range query estimation and in section 5.2 we discuss the higher dimension range query estimation.

5.1 Range query estimation

In this section we discuss the experiment results of the method discussed in the chapter 3 and cosine series technique used in the literature. The analysis has been done using C language. Experiments were performed on a PC with 2.67 GigaHtz processor CPU and 1GB memory. Experiments were done on different datasets.

5.1.1 Normalization of data

To make implementation easier, all the data attribute values are normalized to domain (0,1) by considering maximum value *max* and minimum value *min*.

$$x = \begin{cases} 0, & x \leq min, \\ \frac{x-min}{max-min} & min < x < max, \\ 1 & x \geq max \end{cases} \tag{5.1}$$

We compare the accuracy of selectivity estimation. The technique has been tested for various range queries such as $a \leq X \leq b$. Result has been verified and analyzed for varying number of coefficients and clusters. To perform experiment data sets are normalized to (0,1). Error was calculated as $Error = \frac{Estimatedvalue - Actualvalue}{Actualvalue}$

Table 5.1 presents results of experiments performed on dataset1 having 3769 data values and experiment was conducted using 16 micro-clusters. The synthetic.control.data is synthetic time series dataset. It contains six different classes of control charts i.e. Normal, Cyclic, Increasing trend, Decreasing trend, Upward shift, Downward shift. The results of experiments conducted on dataset2 (synthetic.control.data) having 36025 data values have been shown in table 5.2. Experiments results for dataset3 (docbyterm.tfidf.txt) are shown in table 5.3 and results for dataset4 (ECG dataset mitdbx mitdbx 108) are compared in figure 5.3. Experiments were conducted on dataset5 (taken from advertising dataset who rated what 2006.txt) for different number of clusters and results are shown in table 5.4. For 100 coefficients and different set of queries 45% and 0% of total queries were within the 4% error range and for 200 number of coefficients, 45% and 0 queries were not more than 4% error range. Experiments were conducted on dataset6 (ann gun centroidA.txt), dataset7(ECG dataset chfdbchf15.txt) and dataset8(Respiration dataset nprs43.txt).

TABLE 5.1: Comparison of result for different number of coefficients for synthetic dataset1

Error range	Number of coefficients														
	Micro-cluster based method							Cosine based method							
	100	200	300	400	500	600	700	800	100	200	300	400	500	600	700	800
0-4	25	20	20	25	20	25	25	20	16	16	20	20	16	25	20	33
0-8	33	25	29	29	25	29	29	29	20	25	20	25	25	37	29	37
0-12	33	25	33	33	29	29	29	29	20	29	29	29	33	41	33	37
0-16	37	37	37	37	37	37	37	37	20	29	29	45	33	41	41	41
0-20	37	37	41	41	37	37	41	41	20	37	33	45	33	50	41	45
0-24	41	45	45	45	41	45	45	45	20	37	33	50	37	54	45	54

Table 5.2: Comparison of result for different number of coefficients for synthetic dataset2

% Error	Number of Coefficients															
	Micro-cluster based selectivity estimation								Cosine series method							
1	100	200	300	400	500	600	700	800	100	200	300	400	500	600	700	800
0-4	41	41	50	50	50	50	50	50	0	0	0	0	0	0	0	0
0-8	58	50	58	62	58	58	58	62	0	0	0	0	0	0	0	0
0-12	70	66	70	70	66	70	70	70	0	0	0	0	0	0	0	0
0-16	75	70	70	75	75	75	75	75	0	0	0	0	0	0	0	0
0-20	79	75	75	79	79	79	79	79	0	0	0	0	0	0	0	0
0-24	79	75	75	79	79	79	79	79	0	0	0	0	0	0	0	0

Table 5.3: Comparison of result for different number of coefficients for synthetic dataset3

% Error	Number of Coefficients															
	Micro-clustering based method								Cosine series method							
	100	200	300	400	500	600	700	800	100	200	300	400	500	600	700	800
0-4	45	50	58	58	58	58	58	62	0	0	4	8	0	8	4	8
0-8	50	58	62	62	58	66	62	62	0	4	4	8	4	8	4	8
0-12	70	62	66	70	70	70	70	70	0	8	8	8	8	8	8	8
0-16	79	70	75	79	75	75	75	75	0	12	8	8	12	8	12	8
0-20	79	75	75	79	79	79	79	79	0	12	8	8	12	8	12	8
0-24	79	75	75	79	79	79	79	79	0	12	12	8	16	8	12	8

TABLE 5.4: Comparison of result for different number of coefficients for synthetic dataset5

% Error	Number of Clusters																									
	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34										
0-4	45	37	45	29	45	45	45	50	20	20	33	29	33	29	29	62										
0-8	66	62	70	50	54	50	58	62	41	41	45	45	45	45	45	66										
0-12	66	62	70	54	58	58	62	62	58	62	66	66	62	62	62	66										
0-16	75	70	79	66	62	66	70	75	62	62	66	66	62	66	66	75										
0-20	79	70	83	79	66	70	79	75	62	66	66	66	66	66	66	79										
0-24	79	70	83	79	66	70	79	75	75	75	70	70	70	70	70	83										

TABLE 5.5: Comparison of result for different number of coefficients for synthetic dataset6

% Error	Number of Coefficients														
	Micro-cluster based method							Cosine series method							
	100	200	300	400	500	600	700	800	100	200	300	400	500	600	700	800
0-4	25	25	29	29	20	25	33	33	4	4	4	4	4	4	4	4
0-8	37	41	45	45	45	41	41	41	4	4	4	4	4	4	4	4
0-12	45	50	58	58	54	54	58	54	4	4	4	4	4	4	4	4
0-16	50	54	58	58	58	58	58	58	4	4	4	4	4	4	4	4
0-20	54	54	58	58	58	58	58	58	4	4	4	4	4	4	4	4
0-24	58	62	62	62	62	62	66	62	8	8	8	8	8	8	8	8

TABLE 5.6: Comparison of percentage of queries for given error range and number of values for 12 clusters

% Error	Dataset1	Dataset7	Dataset2	Dataset6	Dataset4	Dataset5	Dataset3
	3793	15024	18077	22527	64825	200025	741502
0-4	20	25	16	25	37	45	54
0-8	25	41	16	41	50	54	62
0-12	29	50	29	50	54	58	66
0-16	33	54	37	54	62	62	70
0-20	37	58	37	54	70	66	75
0-24	45	58	41	58	75	66	75

The proposed technique improves the selectivity estimation of range query over data streams. Results of experiments in tables 5.2, 5.3 shows that there is a significant improvement over cosine series method for different datasets. The result of synthetic dataset1 has got comparable results with existing technique. All the data values are represented compactly in the form of micro-clusters. The spread of data values covered in a micro-cluster is less and the cosine coefficients are used for selectivity estimation. Cosine series method covers the whole range of data values and RQE method works over a narrow range of data values. Hence for narrow range queries RQE technique works better than cosine series technique [118]. The results for RQE and cosine series techniques are comparable if the range of the query covers whole range of data values.

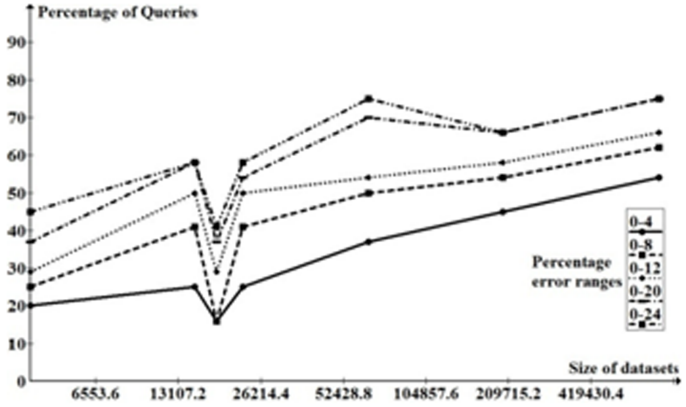

FIGURE 5.1: Comparison of results for increasing size of datasets (12 clusters,200 coefficients)

RQE technique performs well for normal, moderately skewed and highly skewed data distributions. Dataset4 and dataset7 are highly skewed with values 1.66

and -2.41 respectively. Dataset1,dataset3 and dataset5 are moderately skewed with values 0.66,0.879 and 0.938 respectively. Dataset2 and dataset6 are approximately symmetric. Figure 5.1 and 5.2 shows that the results for all these datasets are comparable.

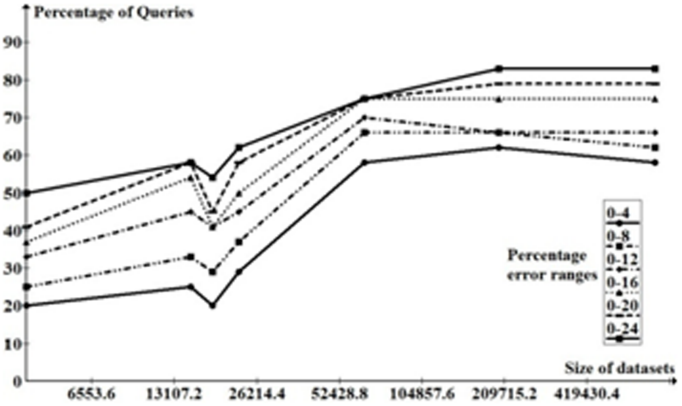

FIGURE 5.2: Comparison of results for increasing size of datasets (36 clusters,200 coefficients)

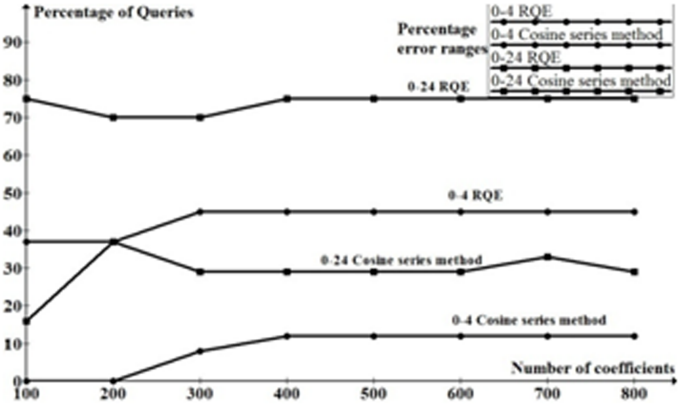

FIGURE 5.3: Comparison of results for dataset4 for 16 clusters

The technique works well with datasets of different sizes. Figure 5.1 and figure 5.2 show that the accuracy of result improves with the increase in the size of dataset. For example data set with size 3793, 22527, 64825 and 741502, percentage of queries within 4% error range are 20%, 25%, 37% and 54% respectively. This means that highest size dataset has highest number of queries in the lowest error range. Cosine series has infinite number of coefficients. It stores significant information in few low frequency coefficients. Accuracy of results improves with the increase in number of stored cosine coefficients. Larger numbers of coefficient

are required for multi- dimensional data. RQE stores cosine coefficients separately for each micro-cluster, so with increase in the number of micro-clusters the requirement of is also increased. On the other hand cosine series require memory to store only one set of cosine coefficients. Processing time of RQE method is more as compared to cosine series method .It has to maintain micro-clusters and search the nearest micro-clusters for selectivity estimation. As the number of clusters increases the more processing time is required for searching the nearest cluster. It has been observed that the small number of clusters also gives comparable result to large no of clusters. The proposed method requires an efficient and accurate clustering algorithm. Better clustering algorithm can improve the efficiency and accuracy of results.

5.2 Rectangular query estimation

In this section, we discuss the experimental results of comparison of our method with existing cosine series method. The experiments were performed on a PC with 2.67 GigaHtz processor CPU and 2GB memory using C language. We compare the estimation accuracy of the rectangular range queries for various datasets. To perform experiments, datasets are normalized to (0,1). The method was tested for 53 different queries such as $a \leq x \leq a_1$ and $b \leq y \leq b_1$. Examples of queries are $(0.01 \leq x \leq 0.10$ and $0.03 \leq y \leq .70), (.01 \leq x \leq .26$ and $0.07 \leq y \leq 0.88), (.06 \leq x \leq .095$ and $.28 \leq y \leq .92), (0.01 \leq x \leq .02$ and $.03 \leq y \leq .4), (.01 \leq x \leq .17$ and $.01 \leq y \leq .51)$.

Results were verified and analyzed for varying number of coefficients. Figure 5.5 presents results of experiments performed on dataset9(household power consumption dataset) having 1076595 records and experiments were conducted for 4 micro-clusters and 100 coefficients. Figure 5.6 shows result of experiments on dataset9 for varying number of coefficients. The results of experiments conducted on dataset7 (ECG dataset chfdbchf15.txt) having 14998 data values has been shown

in Figure 5.7 and Table 5.7. Experiments were conducted on dataset10(docword.nips.txt) are shown in Table 5.8. Size is the number of data values taken in experiment. Percentage of queries showing improved results (PIQ) as comparison to existing method was calculated as:

$$PIQ = \frac{((TQ - QIR) * 100)}{TQ} \tag{5.2}$$

where TQ, QIR are total number of queries and number of queries with improved results respectively. The percentage reduction in average absolute error (aae) i.e. PRE was calculated as:

$$PRE = \frac{((aae_{existingmethod} - aae_{proposedmethod}) * 100)}{aae_{existingmethod}} \tag{5.3}$$

Table 5.7: Results for dataset10

Number of data values	100 coefficients		200 coefficients	
	PIQ(%)	PRE(%)	PIQ(%)	PRE(%)
74631	71.15	83.72	71.2	83.68
149262	90.38	69.90	90.4	69.88
223893	88.46	61.75	88.5	61.73

Table 5.8: Results for different number of data values of dataset7

Number of data values	100 coefficients		200 coefficients		300 coefficients	
	PIQ(%)	PRE(%)	PIQ(%)	PRE(%)	PIQ(%)	PRE(%)
2146	61.5	16.6	61.5	24.7	61.5	24.1
6430	76.8	33.4	76.9	36.9	76.9	36.5
10714	78.8	40.9	78.8	43.5	78.8	42.9
14998	78.8	46.9	78.8	48.8	78.8	48.3

As explained in chapter 1 it is not possible to apply normal selectivity estimation methods on large and continuous data streams. The proposed method estimate selectivity of data stream. It improves the rectangular range query estimation for data streams over cosine series method. The proposed method maintains all the data values compactly in the form of micro-clusters. These micro-clusters are created and maintained dynamically. The method does not require reconstruction of micro-clusters periodically. When the number of micro-clusters reaches maximum, the micro-cluster with oldest timestamp below a threshold is eliminated.

So micro-clusters with recent time stamps are considered for selectivity estimation. These micro-clusters store cosine coefficients, which are used for selectivity estimation. On the other hand, cosine series method stores cosine coefficient over the whole range of data values. Hence, for narrow range queries our method works better than cosine series method. The results for our method and cosine series method are comparable if the range of the query covers whole range of data values. Accuracy of results improves with the increase in number of stored cosine coefficients. A large number of cosine coefficients are required to represent multi-dimensional data values. We store cosine coefficients separately in each micro-cluster, so with increase in the number of micro-clusters increases the storage space, while [118] require memory to store only one set of cosine coefficients. The proposed method require more time to find selectivity of rectangular query as compared to existing method, as it has to maintain micro-clusters and search for the nearest micro-clusters for selectivity estimation. Improved clustering algorithm can improve the efficiency and accuracy of results. The skewness of the data distribution is more for higher dimensions, therefore, the error rates increase. Figure 5.5, 5.6 show result of experiment performed on electric power consumption in one household with a one-minute sampling rate over a period of almost 4 years. It is a Time-Series dataset with real data values. Experiments have been conducted on two fields global active power and global reactive power of this dataset. Table 5.7 shows result of experiment on ECG dataset and table 5.8 shows result of experiments on Bag of Words dataset. The method works well for different types of datasets. We find the average result size of 54 different size queries and estimate the size of the queries with different number of cosine coefficients. The decrease in query size increases error rates. Results are consistent for different sizes of datasets. There is a significant reduction in average absolute error for different sizes of datasets (14998, 1076595). Result of experiments on household power consumption shows that there is significant improvement in the selectivity for varying sizes of data values. The results improve with the increasing size of inputs for different datasets. Figure 5.5 shows that for increasing

number of data sizes 215319, 430638, 645957, 861276, 1076595 there is increase in percentage of queries showing improved results i.e. 84.6, 86.5, 90.4, 94.2, 94.2. Percentage improvement in accuracy of values shows similar trends.

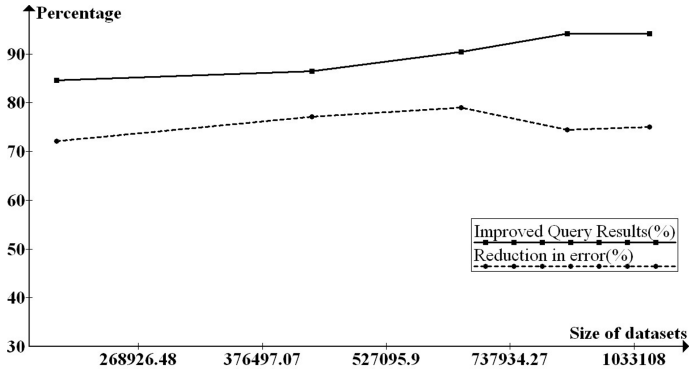

FIGURE 5.4: Percentage of Improvement in results for dataset9.

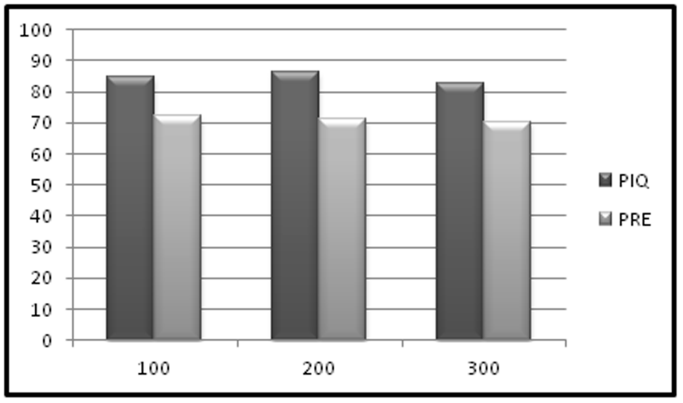

FIGURE 5.5: Improvements for different number of cosine coefficients for dataset9(size 215319)

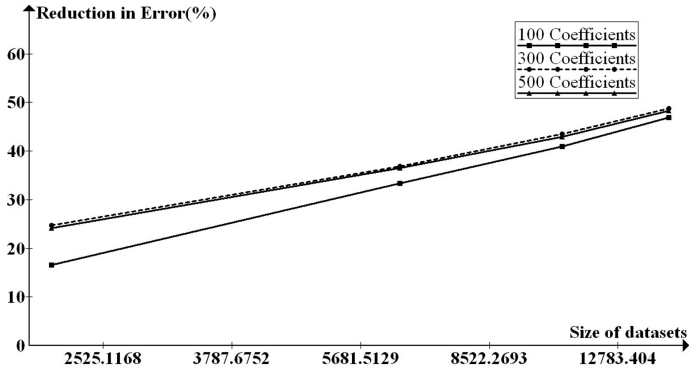

FIGURE 5.6: Reduction in error for dataset7

The high-dimensional Euclidean spaces have a curse of dimensionality which is manifestation of the curse in high dimensions. All pairs of points are equally

far away from one another. Also we observe that almost any two vectors are orthogonal. All points will have a distance close to the average distance. If there are essentially no pairs of points that are close, it is hard to build clusters at all. There is little justification for grouping one pair of points and not another. It is difficult to find clusters from random data when so many pairs that are all at approximately the same distance. The data distribution has impacts on the error rates for estimating the selectivity. As the dimension increase, the skewness of the dataset also increases exponentially and hence the error rates. The skewness of the normal and the clustered distribution increases very slightly with the increase in dimension.

With this discussion we have completed the experimental evaluation of our techniques. In the next chapter, we conclude this work with a summary of its salient results and discussion on future research.

Chapter 6

Conclusions and Future work

This chapter concludes this thesis. In Section 6.1, we briefly summarize our main results for query estimation over data streams using micro-clustering. In Section 6.2, we give an outlook on future research directions, which could use our estimation method for further development.

6.1 Conclusions

The increasing number continuous, unbounded data stream generated by real-world applications necessitates the development of data stream mining techniques. In this thesis, we investigated query estimation, which serves as an important part of data mining applications. There are various approaches in the literature for capturing the essence of a data distribution. In this work, we investigated how to use micro-clustering and cosine series techniques for query estimation over continuous, unbounded and evolving data streams. We introduced a novel approach for query estimation. Our technique compressed the data stream summary in micro-clusters. It scans the data in one pass and updates the statistics incrementally. We used this summary statistics to estimate

queries of stream clusters. We use compaction property of cosine series to approximate the estimation efficiently. The result of estimation over the clusters is aggregated to give the result over a range query. We apply this technique for rectangular queries as well.

We conducted an experimental study for real-world data streams originating from different application scenarios as well as for synthetic streams. The experiments were designed to evaluate the efficacy and efficiency our method with respect to estimation quality, processing time, and memory allocation. The results of the experiments revealed that our technique delivered estimates accurately in comparison to referential method. We conclude that the proposed micro-cluster based technique succeeds in estimating the single dimension range queries and rectangular queries over evolving data streams.

6.2 Future Work

We can use the micro-cluster based selectivity estimation technique in different data mining application for further investigations. The extensions of the technique can be applied to estimate queries over higher dimensions. The high dimension data streams has been challenging problem for large datasets. This is due to that fact that as the dimension increases the computation requirements increase exponentially. Moreover it takes quite a large space to store the summary of multi-dimensional data streams. As discussed in the thesis the work can be extended to higher dimensions. As we store micro-cluster snapshots at different time granularity, we can study the applicability of technique in various applications which require estimation over the past data. Many applications need to estimate queries over recent time window only. The technique can be extended to estimate query over sliding windows. We can use cosine coefficients with hierarchical clustering technique(BIRCH) for query estimation in distributed

environment. Recently lot of research has been done in the field of sensor networks. One can study the applicability of the technique in the field of sensor network.

Appendix A

Publications

- Sudhanshu Gupta, Deepak Garg,"Selectivity Estimation of range queries in Data Streams using Micro-clustering", International Arab Journal of Information and Technology, SCI Indexed, Impact factor 0.39
- Sudhanshu Gupta, Deepak Garg, "Selectivity estimation over multiple data streams using micro-clustering", International Journal of Advancements in Technology (IJoAT),Volume 4,Issue 2, 2014.
- Sudhanshu Gupta, Deepak Garg, "Survey on Query Estimation over Data Streams", IEEE International Advance Computing Conference (IACC 2009), pages(1417-1422), 6-7 March 2009.

Appendix B

Queries used in experiment

Table B.1: Rectangular queries used in experiment

	x	***x1***	***y***	***y1***
1	0.01	.10	.03	.70
2	.01	.26	.07	.88
3	.06	.095	.28	.92
4	.03	.35	.04	.53
5	.06	.7	.08	.9
6	.018	.1	.29	.65
7	0.01	.02	.03	.4
8	.05	.16	.10	.48
9	.09	.095	.098	.82
10	.03	.35	.45	.53
11	.6	.7	.75	.9
12	.038	.386	.09	.95
13	0.01	.70	.03	.84
14	.05	.86	.17	.98
15	.06	.19	.09	.32
16	.03	.13	.05	.53
17	.06	.47	.10	.78
18	.082	.18	.09	.95

TABLE B.2: Rectangular queries used in experiment.. 2

	x	***x1***	***y***	***y1***
19	0.05	.20	.13	.24
20	.01	.76	.05	.9
21	.09	.45	.07	.92
22	.03	.23	.45	.53
23	.06	.17	.57	.70
24	.082	.28	.39	.49
25	.01	.17	.01	.51
26	.1	.7	.01	.92
27	.05	.25	.015	.25
28	.02	.1	.02	.8
29	.05	.3	.25	.35
30	.3	.35	.3	.40
31	.03	.4	.3	.9
32	.35	.45	.35	.45
33	.4	.45	.4	.45
34	.1	.57	.45	.58
35	.04	.75	.045	.55
36	.05	.55	.5	.55

TABLE B.3: Rectangular queries used in experiment .. 3

	x	***x1***	***y***	***y1***
37	.05	.6	.5	.6
38	.15	.65	.55	.65
39	.06	.8	.006	.7
40	.065	.7	.65	.7
41	.065	.8	.65	.8
42	.07	.85	.004	.9
43	.05	.8	.055	.008
44	.075	.25	.75	.85
45	.075	.9	.045	.109
46	.08	.5	.050	.085
47	.08	.1	.05	.9
48	.08	.99	.08	.99
49	.09	.095	.0059	.95
50	.06	.1	.5	.9
51	.06	.08	.056	.8
52	.06	.99	.06	.99
53	.02	.5	.002	.5
54	.1	.9	.001	.9

Appendix C

Query distribution

TABLE C.1: Query distribution percentage of various datasets

	dataset1	**dataset8**	**dataset7**	**dataset6**	**dataset4**	**dataset2**	**dataset5**	**dataset3**
Size	**3769**	**18077**	**15024**	**22527**	**64825**	**36025**	**200025**	**741502**
less than 2%	63	71	63	54	29	21	38	63
2% to 5%	0	21	13	13	25	17	25	25
5% to 10%	8	0	4	8	17	13	17	4
10% to 20%	29	0	8	4	13	17	21	8
>20%	0	8	13	21	17	33	0	0

Bibliography

[1] Mukesh Kumar Jha and T. P. Sharma. Secure data aggregation in wireless sensor network: a survey. *International Journal of Engineering Science and Technology*, 3, 2011.

[2] Charu C. Aggarwal, editor. *Data Streams: Models and Algorithms*. Springer, 2007.

[3] P. Domingos and G. Hulten. A general framework for mining massive data streams. *Journal of Computational and Graphical Statistics*, 12, 2003.

[4] M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining data streams: a review. In *SIGMODRecord*, volume 34, pages 18–26, 2005.

[5] Jiawei Han and Micheline Kamber. *Data Mining Concepts and Techniques*. Elsevier, 2001.

[6] M. Mohoui, B. Bhargava, and M. Mohania. Data mining for web security. In *Proceedings of IC'2001*, 2001.

[7] Zhao Peixiang, Charu C. Aggarwal, and Min Wang. gsketch: On query estimation in graph streams. In *Proceedings of the VLDB Endowmen*, volume 5, pages 193–204, 2012.

[8] Chih-Ping Wei, Selwyn Piramuthu, and Michael J. Shaw. *Clyde W. Holsapple (ed.) Handbook on Knowledge Management*, volume 2, chapter Knowledge Discovery and Data Mining, pages 157–189. Berlin/Heidelberg: Springer-Verlag, 2002.

[9] Alaa Aljanaby, Emad Abuelrub, and Mohammed Odeh. A survey of distributed query optimization. *The International Arab Journal of Information Technology*, 2:48–57, 2005.

[10] S. Muthukrishnan. Data streams: Algorithms and applications. In *Proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms.*, pages 413–413, 2003.

[11] A. Arasu, B. Babcock, S. Babu, J. McAlister, and J. Widom. Characterizing memory requirements for queries over continuous data streams. In *In Proceeding of the 2002 ACM Symp. on Principles of Database Systems*, 2002.

[12] Abhsihek Kumar, Minho Sung, Jun Jim Xu, and Jia Wang. Data streaming algorithms for efficient and accurate estimation of flow size distribution. In *SIGMETRICS '04/Performance '04 Proceedings of the joint international conference on Measurement and modeling of computer systems*, pages 177–188, 2004.

[13] Ezio Lefons, Alberto Silvestri, and Fillippo Tangorra. An analytic approach to statistical databases. In *In Proceedings of the 9th International Conference on Very Large Data Bases*, pages 260–274, 1983.

[14] Michael V. Mannino, Paicheng Chu, and Thomas Sagar. Statistical profile estimation in database systems. *ACM Comput.Surv.*, 20:191–221, 1988.

[15] Wei Sun, Yibei Ling, and Naphtali Deng Yi Rishe. An instant and accurate size estimation method for joins and selections in a retrieval-intensive environment. In *J SIGMOD Rec.*, volume 22, pages 79–88, 1993.

[16] D. Datar, A. Gionis, P. Indyk, and R. Motwan. Maintaining stream statistics over sliding windows. In *In Proceedings of the ACM-SIAM Symposium on Discrete Mathematics(SODA),*, pages 635–644, 2002.

[17] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classiffiers. In *In Proceedings of the ACM SIGKDD International Conference onKnowledge Discovery and Data Mining*, pages 226–235, 2003.

[18] C. Aggarwal, J. Han, J. Wang, and P. Yu. On demand classifcation of data streams. In *In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[19] C Aggarwal. A framework for change diagnosis of data streams. In *In Proceedings ofthe ACM SIGMOD International Conference on Management of Data*, pages 575–586, 2003.

[20] S. Ben-David, J. Gehrke, and D. Kifer. Detecting change in data streams. In *In Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 180–191, 2004.

[21] G. Dong, J. Han, L. V. S. Lakshmanan, J. Pei, and H. Yu P. S. Wang. Online mining of changes from data streams: Research problems and preliminary results. In *n Proceedingsof the ACM SIGMOD Workshop on Management and Processing of Data Streams*, 2003.

[22] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,*, pages 97–106, 2001.

[23] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceeding PODS '02 Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16, 2002. doi: 10.1145/543613.543615.

[24] G. Hulten and P. Domingos. Mining high-speed data streams. In *In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,*, pages 71–80, 2000.

[25] Nick Koudas and D. Shrivastava. Data stream query processing: A tutorial. In *In Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 2003.

[26] Lukasz Golab and M. Tamer zsu. Issues in data stream management. In *ACM SIGMOD*, volume 32, pages 5–14, 2003. doi: 10.1145/776985.776986.

[27] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and Varm. Query processing, approximation, and resource management in a data stream management system. In *In Proceedings of the InternationalConference on Innovative Data Systems Research (CIDR)*, 2003.

[28] Amol Deshpande, Zachary Ives, and Vijayshankar Raman. Adaptive query processing. *Foundations and Trends in Databases*, 1:1–140, 2007.

[29] R. Lipton, J. Naughton, and D. Schneider. Practical selectivity estimation through adaptive sampling. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, volume 19, pages 1–11, 1990. doi: 10.1145/93597.93611.

[30] G. Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, 25:73–170, 1993.

[31] M. Garofalakis, J. Gehrke, and R. Rastogi. Querying and mining data streams you only get one look. In *In Proceedings of the International Conference on Very Large DataBases (VLDB)*, 2002.

[32] S. Schmidt, T. Legler, S. Schär, and W. Lehner. Robust real-time query processing with qstream. In *In Proceedings of the International Conference on Very Large DataBases (VLDB),*, pages 1299–1302, 2005.

[33] Y. Auvil, D. Cai, D. Clutter, J. Han, G. Pape, and M. Welge. Maids : Mining alarming incidents from data stream. In *In Proceedings of the ACM*

SIGMOD InternationalConference on Management of Data, pages 919–920, 2004.

[34] A. Manjhi, V. Shkapenyuk, Dhamdhere, and C. Olston. Finding (recently) frequent items in distributed data streams. In *In Proceedings of the IEEE International Conferenceon Data Engineering (ICDE),*, pages 767–778, 2005.

[35] C. D. Johnson T. Spatscheck O. Shkapenyuk. V Cranor. Gigascope:a stream database for network applications. In *In Proceedings of the ACM SIGMOD InternationalConference on Management of Data*, pages 647–651, 2003.

[36] C. D. Cranor, T. Johnson, O. Spatscheck, and Shkapenyuk.V. The gigascope stream database. *IEEE Data Engineering Bulletin,*, 26:27–32, 2003.

[37] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *In Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 358–369, 2002.

[38] H. Kargupta, R. Bhargava, K. Liu, M. Powers, P. Blair, S. Bushra, J. Dull, K. Sarkar, M. Lein, M. Vasa, and D. Handy. Vedas: A mobile and distributed data stream mining system for real-time vehicle monitoring. In *In Proceedings of the SIAM InternationalConference on Data Mining (SDM)*, 2004.

[39] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. Hellerstein, W. Hong, S. Krishnamurthy, S. R. V. Madden, F. Reiss Raman, and M. A. Shah. Telegraphcq: Continuous data processing for an uncertain world. In *In Proceedings ofthe ACM SIGMOD International Conference on Management of Data*, 2003.

[40] S. Krishnamurthy, S. Chandrasekaran, O. Cooper, A. Deshpande, Franklin M. J., J. Hong W. Hellerstein, S. Reiss F. Madden, and M. A. Shah. Telegraphcq: An architectural status report. *IEEE Data Engineering Bulletin*, 26:11–18, 2003.

[41] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *In SDM Conference*, 2006.

[42] W. Fan, Y. Huang, H. Wang, and P.S. Yu. Active mining of data streams. In *In Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2004.

[43] V. Ganti, J. Gehrke, and R. Ramakrishnan. Demon: Mining and monitoring evolving data. *IEEE Transactions on Knowledge and Data Engineering (TKDE),*, 13:50–63, 2002.

[44] A. Arasu, B. Babcock, and J. Widom. Cql :the continuous query language. *The VLDB Journal*, 2005.

[45] J. Kramer and B. Seeger. A temporal foundation for continuous queries over data streams. In *In Proceedings of the International Conference on Management of Data (COMAD)*, pages 70–82, 2003.

[46] Abhinandan Das, Johannes Gehrke, and Mirek Riedewald. Approximation techniques for spatial data. In *In Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 695–706, 2004.

[47] Nikos Mamoulis and Papadias Dimitris. Selectivity estimation of complex spatial queries. In *In Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, pages 155–174, 2001.

[48] Charu C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th VLDB Conference*, pages 81–92, 2003.

[49] C. Aggarwal and C. Reddy, editors. *Data Clustering: Algorithms and Applications*, chapter A Survey of Stream Clustering Algorithms. CRC Press, 2013.

[50] J. Hüllermeier E. Beringer. Online clustering of parallel data streams. *Data Knowledgeand Engineering*, 58:180–204, 2006.

[51] Chris Bohem, Hans-Peter Kriegel, Peer Kroger, and Petra Linhart. Selectivity estimation of high dimensional window queries via clustering. In *Proceedings of the 9th international conference on Advances in Spatial and Temporal Databases*, pages 1–18, 2005. doi: 10.1007/11535331_1.

[52] M. Charikar, O. L. Callaghan, and R. Panigrahy. Better streaming algorithms for clustering problems. In *In Proceedings of the ACM Symposium on Theory of Computing(STOC),*, pages 30–39, 2003.

[53] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *TKDE special issue on clustering*, 15, 2003.

[54] G. Kollios, D. Gunopulos, Nick Koudas, and S. Berchtold. Efficient biased sampling for approximate clustering and outlier detection in large data sets. *Knowledge and Data Engineering (TKDE)*, 15:1170–1187, 2003.

[55] Thomos Likewin and B. Annapa. Application of parallel k-means clustering algorithm for prediction of optimal path in self aware mobile ad-hoc networks with link stability. *Advances in Computing and Communications*, pages 396–405, 2011.

[56] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high-quality clustering. In *In Proceedings of the IEEE International Conferenceon Data Engineering (ICDE)*, 2002.

[57] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch:an efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data - SIGMOD '96*, pages 103–114, 1996.

[58] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, R. Motwani, I. Nishizawa, U. Srivastava, D. Thomas, R. Varma, and J. Widom. Stream:the stanford stream data manager. *IEEE Data Engineering Bulletin*, 26:19–26, 2003.

[59] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. The clustree: Indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems*, 29(2):249–272, 2011.

[60] Y. Chen and L. Tu. Density-based clustering for real time stream data. In *ACMKDD Conference*, 2007.

[61] J. Gao, J. Li, Z. Zhang, and P.-N. Tan. An incremental data stream clustering algorithm based on dense units detection. In *In PAKDD Conference*, 2005.

[62] C. Aggarwal, J. Han, J. Wang, and P. Yu. On high dimensional projected clustering of data streams. *Data Mining and Knowledge Discovery Journal*, 10(3):251–273, 2005.

[63] Z. He, X. Xu, S. Deng, and J. Huang. Clustering categorical data streams. *Arxiv*, 2004.

[64] C. Aggarwal. .a framework for clustering massive-domain data streams. In *In ICDE Conference*, 2009.

[65] Graham Cormode and S. Muthukrishnan. What's hot and what's not: tracking frequent items dynamically. In *In Proceedings of the ACM Symposium on Principles of DatabaseSystems (PODS),*, pages 296–306, 2003.

[66] G. Cormode, S. Muthukrishnan, and W. Zhuang. Conquering the divide: Continuous clustering of distributed data streams. In *ICDE Conference*, 2007.

[67] Sudipto Guha, Nick Koudas, and Kyuseok Shim. Approximation and streaming algorithms for histogram construction problems. *ACM Trans. Database Syst*, 31, 2006.

[68] M. Muralikrishna and D. J. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *In Proceedings of*

the 1988 ACM International Conference on Management of Data (SIGMOD'88), pages 28–36, 1988.

[69] H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, K. C. Sevcik, and T. Suel. Optimal histograms with quality guarantees. In *Proceedings of the 24th International Conference on Very Large Data Bases*, pages 275–286, 1998.

[70] Sudipto Guha, N. Koudas, Shim, and . Data-streams and histograms. In *In Proceedings ofthe ACM Symposium on Theory of Computing (STOC)*, pages 471–476, 2001.

[71] A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Fast small-space algorithms for approximate histogram maintenance. In *In Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 389–398, 2002.

[72] Graham Cormode and S. Muthukrishnan. An improved data stream summary:the count-min sketch and its applications. *Journal of Algorithms*, 55: 58–75, 2005. doi: 10.1016/j.jalgor.2003.12.001.

[73] Li.T., Q. Li, S. Zhu, and M. Ogihara. A survey on wavelet applications. In *SIGKDD Explorations*, volume 4, pages 49–68, 2002.

[74] P. Karras and N. Mamoulis. One-pass wavelet synopses for maximum-error metrics. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 421–432, 2005.

[75] S. Guha and B. Harb. Wavelet synopsis for data streams: minimizing non-euclidean error. In *In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 88–97, 2005.

[76] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *In Proceedings of the 2002 Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 633–634, 2002.

[77] Gurmeet Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of VLDB '02 Proceedings of the 28th international conference on Very Large Data Bases*, pages 346–357, 2002.

[78] S. Guha, P. Indyk, N. Koudas, and N. Thaper. Dynamic multidimensional histograms. In *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 428–439, 2002.

[79] Flip Korn, Theodore Johnson, and H. V. Jagadish. Range selectivity estimation for continuous attributes. In *Proceedings of the SSDM Conference*, pages 244–253, 1999.

[80] C. Aggarwal. On biased reservoir sampling in the presence of stream evolution. In *In Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 607–618, 2006.

[81] Chunmin Melvin Chen and Nick Roussopoulos. Adaptive selectivity estimation using query feedback. In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, volume 23, pages 161–172, 1994. doi: 10.1145/191843.191874.

[82] Peter J. Haas and Arun N. Swami. Sequential sampling procedures for query size estimation. In *Proceedings of the ACM SIGMOD Conference*, pages 341–350, 1992.

[83] Phillip B. Gibbons and Yossi Matias. New sampling based summary statistics for improving approximate query answers. In *SIGMOD '98 Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, volume 27, pages 331–342, 1998. doi: 10.1145/276304.276334.

[84] Yibei Ling and Wei Sun. A supplement to sampling-based methods for query size estimation in a database system. In *SIGMOD Rec.*, volume 21, pages 12–15, 1992.

[85] Yi-Leh Wu, D. Agrawal, and Amr El Abbadi. Applying the golden rule of sampling for query estimation. In *Proceeding of ACM SIGMOD international conference on management of data*, pages 449–460, 2001.

[86] Yi-Leh Wu, D. Agrawal, and Amr El Abbadi. Query estimation by adaptive sampling. In *Proceedings of the 18th International Conference on data engineering*, pages 639–648, 2002.

[87] Sawrup Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join synopses for approximate query answering. In *Proceedings of the 1999 ACM SIGMOD international conference on management of data*, volume 28, pages 275–286, 1999. doi: 10.1145/304181.304207.

[88] Sawrup Acharya, Viswanath Poosala, and S. Ramaswamy. Selectivity estimation in spatial databases. In *Proceedings of the 1999 ACM SIGMOD international conference on management of data*, volume 28, pages 13–24, 1999. doi: 10.1145/304182.304184.

[89] Sudipto Guha and Nick Koudas. Approximating a data streams for querying and estimation: algorithms and performance evaluation. In *Proceedings of the 18th International Conference on Data Engineering*, pages 567–576, 2002. doi: 10.1109/ICDE.2002.994775.

[90] Yannis E. Ioannidis and Viswanath Poosala. Balancing histogram optimality and practicality for query result size estimation. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, volume 24, pages 233–244, 1995. doi: 10.1145/223784.223841.

[91] Viswanath Poosala, Yannis E. Ioannidis, Peter J. Haas, and Eugene J. Shekita. Improved histogram for selectivity estimation of range predicates. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, volume 25, pages 294–305, 1996. doi: 10.1145/235968.233342.

[92] Viswanath Poosala and Yannis E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *Proceedings of the 23rd International Conference on Very Large Data Bases Proceedings of the 23rd VLDB Conference*, pages 486–495, 1997.

[93] Ju-Hong Lee, Deok-Hwan Kim, and Chung Chung. Multi-dimensional selectivity estimation using compressed histogram information. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, volume 28, pages 205–214, 1999. doi: 10.1145/304182.304200.

[94] N. Bruno, S. Chaudhuri, and L. Gravano. Stholes: a multidimensional workload-aware histogram. In *Proceedings of the 2001 ACM SIGMOD international conference on management of data*, volume 30, pages 211–222, 2001. doi: 10.1145/376284.375686.

[95] Dennis Fuchs, Zhen He, and Suk Byung Lee. Compressed histograms with arbitrary bucket layouts for selectivity estimation. *Information Sciences*, 177:680–702, 2007.

[96] Francesco Buccafurri, Luigi Pontieri, Domenico Rosaci, and Domenico Sacca. Improving range query estimation on histograms. In *Proceedings of the 18th International Conference on Data Engineering (ICDE.02)*, pages 628–638, 2002.

[97] Francesco Buccafurri, Lax, and Gianluca. Fast range query estimation by n-level tree histogram. *Data & Knowledge Engineering*, 51:257–275, 2004. doi: 10.1016/j.datak.2004.04.004.

[98] Dimitrios Gunopulos, George Kollios, Vassilis J. Tsotras, and Carlotta Domeniconi. Approximating multi-dimensional aggregate range queries over real attributes. In *Proceedings of the ACM SIGMOD Conference*, pages 463–474, 2000.

[99] Dimitrios Gunopulos, George Kollios, Vassilis J. Tsotras, and Carlotta Domeniconi. Selectivity estimators for multidimensional range queries over real attributes. *The VLDB Journal*, 14:137–154, 2005.

[100] Sumit Ganguly, Minos Garofalakis, Amit Kumar, and Rajeev Rastogi. Join-distinct aggregate estimation over update streams. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 259–270, 2005. doi: 10.1145/1065167.1065200.

[101] Anna Gilbert, Yannis Kotidis, S. Muthurkrishan, and Martin Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proceedings of the VLDB Conference*, volume 27, pages 79–88, 2001.

[102] S. S. Ilic and P. Spalevic. Using wavelet packets for selectivity estimation. *The Computer Journal*, 2013.

[103] Yossi Matias, Jeffery Scott Vitter, and Min Wang. Wavelet-based histograms for selectivity estimation. In *SIGMOD '98 Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, volume 27, pages 448–459, 1998. doi: 10.1145/276305.276344.

[104] Y. Matias, J. Vitter, and M. Wang. Dynamic maintenance of wavelet-based histograms. In *Proceeding VLDB '00 Proceedings of the 26th International Conference on Very Large Data Bases*, pages 101–110, 2000.

[105] Kaushik Chakrabarti, Minos Garofalakis, Rajeev Rastogi, and Kyuseok Shim. Approximate query processing using wavelets. *The VLDB Journal*, 10:199–223, 2001.

[106] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for database applications. *Journal of Computer and System Sciences*, 31 (2):182–209, 1985.

[107] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximation the frequency moments. In *Proceedings of 28th Annual ACM Symposium on the Theory of Computing*, pages 20–29, 1996.

[108] Noga Alon, P. B. Phillip, Yossi Matias, and Mario Szegedy. Tracking join and self-join sizes in limited storage. *Computer and System Sciences*, 64: 719–747, 2002.

[109] Alin Dobra, Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. Processing complex aggregate queries over data streams. In *Proceedings of the 2002 ACM SIGMOD International Conferenceon Management of Data*, pages 61–72, 2002. doi: 10.1145/564691.

[110] Alin Dobra, Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. Sketch-based multi-query processing over data streams. In *EDBT*, pages 551–568, 2004.

[111] Sumit Ganguly, Minos Garofalakis, and Rageev Rastogi. Processing data-stream join aggregates using skimmed sketches. In *EDBT*, pages 569–586, 2004.

[112] Anna Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin Strauss. One-pass wavelet decompositions of data streams. *IEEE Transactions on Knowledge and Data Engineering*, 15:541–554, 2003.

[113] Z. Botev, J. Grotowski, and D. Kroese. Kernel density estimation via diffusion. *The Annals of Statistics*, 38:2916–2957, 2010.

[114] C. M. Procopiuc and O. Procopiuc. Density estimation for spatial data streams. In *In Proceedings of the International Symposium on Spatial and Temporal Databases(SSTD)*, pages 109–126, 2005.

[115] Aoying Zhou, Zhiyuan Cai, Li Wei, and Weining Qian. M-kernel merging : Towards density estimation over data streams. In *In Proceedings of the*

International Conference on Database Systemsfor Advanced Applications (DASFAA), volume 198, pages 285–292, 2003.

[116] Christoph Heinz and Bernhard Seeger. Towards kernel density estimation over streaming data. In *Proc. 13th Int'l Conf. Management of Data*, 2006.

[117] Christoph Heinz and Bernhard Seeger. Wavelet density estimators over data streams. In *Proceedings of the ACM Symposium on applied computing*, pages 578–579, 2005.

[118] F. Yan, W. C. Hou, Z. Jiang, C. Luo, and Q. Zhu. Selectivity estimation of range queries based on data density approximation via cosine series. *Data & Knowledge Engineering*, 63:855–878, 2007.

[119] Zhewei Jiang, Cheng Luo, Wen Chi Hou, Feng Yan, Qiang Zhu, and Chih Fang Wang. Join size estimation over data streams using cosine series. *International Journal of Information Technology*, 13:27–46, 2007.

[120] Wen Chi Hou, Chen Luo, Zhewei Jiang, and Feng Yan. Approximate range-sum queries over data cubes using cosine transform. *International Journal of Information and Communication Engineering*, 4:584–590, 2008.

[121] Charu C. Aggarwal. On futurisitic query processing in data streams. In *Proceeding EDBT'06 Proceedings of the 10th international conference on Advances in Database Technology*, 2006. doi: 10.1007/11687238_6.

[122] V. Markl, P. J. Haas, M. Kutsch, N. Megiddo, U. Srivastava, and T. M. Tran. Consistent selectivity estimation via maximum entropy. *The VLDB Journal*, 16:55–76, 2007. doi: 10.1007/s00778-006-0030-1.

[123] Hein To, Kuorong Chiang, and Cyrus Shahabi. Entropy-based histograms for selectivity estimation. In *Proceedings of the 22nd ACM international Conference on information & knowledge management (CIKM'13)*, pages 1939–1948, 2013. doi: 10.1145/2505515.2505756.

[124] Christopher Re and Dan Suciu. Understanding cardinality estimation using entropy maximization. *ACM Transactions on Databases Systems*, pages 1–31, 2012.