Chapter 9

A SURVEY OF SYNOPSIS CONSTRUCTION IN DATA STREAMS

Charu C. Aggarwal
IBM T. J. Watson Research Center
Hawthorne, NY 10532
charu@us.ibm.com

Philip S. Yu
IBM T. J. Watson Research Center
Hawthorne, NY 10532
psyu@us.ibm.com

Abstract

The large volume of data streams poses unique space and time constraints on the computation process. Many query processing, database operations, and mining algorithms require efficient execution which can be difficult to achieve with a fast data stream. In many cases, it may be acceptable to generate *approximate solutions* for such problems. In recent years a number of *synopsis structures* have been developed, which can be used in conjunction with a variety of mining and query processing techniques in data stream processing. Some key synopsis methods include those of sampling, wavelets, sketches and histograms. In this chapter, we will provide a survey of the key synopsis techniques, and the mining techniques supported by such methods. We will discuss the challenges and tradeoffs associated with using different kinds of techniques, and the important research directions for synopsis construction.

1. Introduction

Data streams pose a unique challenge to many database and data mining applications because of the computational and storage costs associated with the large volume of the data stream. In many cases, synopsis data structures

and statistics can be constructed from streams which are useful for a variety of applications. Some examples of such applications are as follows:

- **Approximate Query Estimation:** The problem of query estimation is possibly the most widely used application of synopsis structures [11]. The problem is particularly important from an efficiency point of view, since queries usually have to be resolved in online time. Therefore, most synopsis methods such as sampling, histograms, wavelets and sketches are usually designed to be able to solve the query estimation problem.
- **Approximate Join Estimation:** The efficient estimation of join size is a particularly challenging problem in streams when the domain of the join attributes is particularly large. Many methods [5, 26, 27] have recently been designed for efficient join estimation over data streams.
- **Computing Aggregates:** In many data stream computation problems, it may be desirable to compute aggregate statistics [40] over data streams. Some applications include estimation of frequency counts, quantiles, and heavy hitters [13, 18, 72, 76]. A variety of synopsis structures such as sketches or histograms can be useful for such cases.
- **Data Mining Applications:** A variety of data mining applications such as change detection do not require to use the individual data points, but only require a temporal synopsis which provides an overview of the behavior of the stream. Methods such as clustering [1] and sketches [88] can be used for effective change detection in data streams. Similarly, many classification methods [2] can be used on a supervised synopsis of the stream.

The design and choice of a particular synopsis method depends on the problem being solved with it. Therefore, the synopsis needs to be constructed in a way which is friendly to the needs of the particular problem being solved. For example, a synopsis structure used for query estimation is likely to be very different from a synopsis structure used for data mining problems such as change detection and classification. In general, we would like to construct the synopsis structure in such a way that it has wide applicability across broad classes of problems. In addition, the applicability to data streams makes the efficiency issue of space and time-construction critical. In particular, the desiderata for effective synopsis construction are as follows:

- **Broad Applicability:** Since synopsis structures are used for a variety of data mining applications, it is desirable for them to have as broad an applicability as possible. This is because one may desire to use the underlying data stream for as many different applications. If synopsis construction methods have narrow applicability, then a different structure

will need to be computed for each application. This will reduce the time and space efficiency of synopsis construction.

- **One Pass Constraint:** Since data streams typically contain a large number of points, the contents of the stream cannot be examined more than once during the course of computation. Therefore, all synopsis construction algorithms are designed under a one-pass constraint.

- **Time and Space Efficiency:** In many traditional synopsis methods on static data sets (such as histograms), the underlying dynamic programming methodologies require super-linear space and time. This is not acceptable for a data stream. For the case of space efficiency, it is not desirable to have a complexity which is more than linear in the size of the stream. In fact, in some methods such as sketches [44], the space complexity is often designed to be logarithmic in the *domain-size* of the stream.

- **Robustness:** The error metric of a synopsis structure needs to be designed in a robust way according to the needs of the underlying application. For example, it has often been observed that some wavelet based methods for approximate query processing may be optimal from a global perspective, but may provide very large error on some of the points in the stream [65]. This is an issue which needs the design of robust metrics such as the maximum error metric for stream based wavelet computation.

- **Evolution Sensitive:** Data Streams rarely show stable distributions, but rapidly evolve over time. Synopsis methods for static data sets are often not designed to deal with the rapid evolution of a data stream. For this purpose, methods such as clustering [1] are used for the purpose of synopsis driven applications such as classification [2]. Carefully designed synopsis structures can also be used for forecasting futuristic queries [3], with the use of evolution-sensitive synopsis.

There are a variety of techniques which can be used for synopsis construction in data streams. We summarize these methods below:

- **Sampling methods:** Sampling methods are among the most simple methods for synopsis construction in data streams. It is also relatively easy to use these synopsis with a wide variety of application since their representation is not specialized and uses the same multi-dimensional representation as the original data points. In particular reservoir based sampling methods [92] are very useful for data streams.

- **Histograms:** Histogram based methods are widely used for static data sets. However most traditional algorithms on static data sets require

super-linear time and space. This is because of the use of dynamic programming techniques for optimal histogram construction. Their extension to the data stream case is a challenging task. A number of recent techniques [37] discuss the design of histograms for the dynamic case.

- **Wavelets:** Wavelets have traditionally been used in a variety of image and query processing applications. In this chapter, we will discuss the issues and challenges involved in dynamic wavelet construction. In particular, the dynamic maintenance of the dominant coefficients of the wavelet representation requires some novel algorithmic techniques.

- **Sketches:** Sketch-based methods derive their inspiration from wavelet techniques. In fact, sketch based methods can be considered a randomized version of wavelet techniques, and are among the most space-efficient of all methods. However, because of the difficulty of intuitive interpretations of sketch based representations, they are sometimes difficult to apply to arbitrary applications. In particular, the generalization of sketch methods to the multi-dimensional case is still an open problem.

- **Micro-cluster based summarization:** A recent micro-clustering method [1] can be used be perform synopsis construction of data streams. The advantage of micro-cluster summarization is that it is applicable to the multi-dimensional case, and adjusts well to the evolution of the underlying data stream. While the empirical effectiveness of the method is quite good, its heuristic nature makes it difficult to find good theoretical bounds on its effectiveness. Since this method is discussed in detail in another chapter of this book, we will not elaborate on it further.

In this chapter, we will provide an overview of the different methods for synopsis construction, and their application to a variety of data mining and database problems. This chapter is organized as follows. In the next section, we will discuss the sampling method and its application to different kinds of data mining problems. In section 3, we will discuss the technique of wavelets for data approximation. In section 4, we will discuss the technique of sketches for data stream approximation. The method of histograms is discussed in section 4. Section 5 discusses the conclusions and challenges in effective data stream summarization.

2. Sampling Methods

Sampling is a popular tool used for many applications, and has several advantages from an application perspective. One advantage is that sampling is easy and efficient, and usually provides an *unbiased* estimate of the underlying data with *provable error guarantees*. Another advantage of sampling methods

is that since they use the original representation of the records, they are easy to use with any data mining application or database operation. In most cases, the error guarantees of sampling methods generalize to the mining behavior of the underlying application. Many synopsis methods such as wavelets, histograms, and sketches are not easy to use for the multi-dimensional cases. The random sampling technique is often the only method of choice for high dimensional applications.

Before discussing the application to data streams, let us examine some properties of the random sampling approach. Let us assume that we have a database $\mathcal{D}$ containing N points which are denoted by $X_1 \ldots X_N$. Let us assume that the function $f(\mathcal{D})$ represents an operation which we wish to perform on the database $\mathcal{D}$. For example $f(\mathcal{D})$ may represent the mean or sum of one of the attributes in database $\mathcal{D}$. We note that a random sample S from database $\mathcal{D}$ defines a random variable $f(S)$ which is (often) closely related to $f(\mathcal{D})$ for many commonly used functions. It is also possible to estimate the standard deviation of $f(S)$ in many cases. In the case of *aggregation based* functions in linear separable form (eg. sum, mean), the law of large numbers allows us to approximate the random variable $f(S)$ as a normal distribution, and characterize the value of $f(\mathcal{D})$ probabilistically. However, not all functions are aggregation based (eg. min, max). In such cases, it is desirable to estimate the mean μ and standard deviation σ of $f(S)$. These parameters allows us to design *probabilistic bounds* on the value of $f(S)$. This is often quite acceptable as an alternative to characterizing the entire distribution of $f(S)$. Such probabilistic bounds can be estimated using a number of inequalities which are also often referred to as *tail bounds*.

The markov inequality is a weak inequality which provides the following bound for the random variable X:

$$P(X > a) \leq E[X]/a = \mu/a \tag{9.1}$$

By applying the Markov inequality to the random variable $(X - \mu)^2/\sigma^2$, we obtain the Chebychev inequality:

$$P(|X - \mu| > a) \leq \sigma^2/a^2 \tag{9.2}$$

While the Markov and Chebychev inequalities are farily general inequalities, they are quite loose in practice, and can be tightened when the distribution of the random variable X is known. We note that the Chebychev inequality is derived by applying the Markov inequality on a function of the random variable X. Even tighter bounds can be obtained when the random variable X shows a specific form, by applying the Markov inequality to parameterized functions of X and optimizing the parameter using the particular characteristics of the random variable X.

The Chernoff bound [14] applies when X is the sum of several independent and identical Bernoulli random variables, and has a lower tail bound as well as an upper tail bound:

$$P(X < (1-\delta)\mu) \leq e^{-\mu\delta^2/2} \tag{9.3}$$

$$P(X > (1+\delta)\mu) \leq \max\{2^{-\delta\mu}, e^{-\mu\delta^2/4}\} \tag{9.4}$$

Another kind of inequality often used in stream mining is the Hoeffding inequality. In this inequality, we bound the sum of k *independent bounded* random variables. For example, for a set of k independent random variables lying in the range $[a, b]$, the sum of these k random variables X satisfies the following inequality:

$$P(|X - \mu| > \delta) \leq 2e^{-2k\cdot\delta^2/(b-a)^2} \tag{9.5}$$

We note that the Hoeffding inequality is slightly more general than the Chernoff bound, and both bounds have similar form for overlapping cases. These bounds have been used for a variety of problems in data stream mining such as classification, and query estimation [28, 58]. In general, the method of random sampling is quite powerful, and can be used for a variety of problems such as order statistics estimation, and distinct value queries [41, 72].

In many applications, it may be desirable to pick out a sample (reservoir) from the stream with a pre-decided size, and apply the algorithm of interest to this sample in order to estimate the results. One key issue in the case of data streams is that we are not sampling from a fixed data set with *known size* N. Rather, the value of N is unknown in advance, and the sampling must be performed dynamically as data points arrive. Therefore, in order to maintain an unbiased representation of the underlying data, the probability of including a point in the random sample should not be fixed in advance, but should change with progression of the data stream. For this purpose, reservoir based sampling methods are usually quite effective in practice.

2.1 Random Sampling with a Reservoir

Reservoir based methods [92] were originally proposed in the context of one-pass access of data from magnetic storage devices such as tapes. As in the case of streams, the number of records N are not known in advance and the sampling must be performed dynamically as the records from the tape are read.

Let us assume that we wish to obtain an unbiased sample of size n from the data stream. In this algorithm, we maintain a reservoir of size n from the data stream. The first n points in the data streams are added to the reservoir for initialization. Subsequently, when the $(t+1)$th point from the data stream is received, it is added to the reservoir with probability $n/(t+1)$. In order

to make room for the new point, any of the current points in the reservoir are sampled with equal probability and subsequently removed.

The proof that this sampling approach maintains the unbiased character of the reservoir is straightforward, and uses induction on t. The probability of the $(t+1)$th point being included in the reservoir is $n/(t+1)$. The probability of any of the last t points being included in the reservoir is defined by the sum of the probabilities of the events corresponding to whether or not the $(t+1)$th point is added to the reservoir. From the inductive assumption, we know that the first t points have equal probability of being included in the reservoir and have probability equal to n/t. In addition, since the points remain in the reservoir with equal probability of $(n-1)/n$, the conditional probability of a point (among the first t points) remaining in the reservoir given that the $(t+1)$ point is added is equal to $(n/t) \cdot (n-1)/n = (n-1)/t$. By summing the probability over the cases where the $(t+1)$th point is added to the reservoir (or not), we get a total probability of $((n/(t+1)) \cdot (n-1)/t + (1-(n/(t+1))) \cdot (n/t) = n/(t+1)$. Therefore, the inclusion of all points in the reservoir has equal probability which is equal to $n/(t+1)$. As a result, at the end of the stream sampling process, all points in the stream have equal probability of being included in the reservoir, which is equal to n/N.

In many cases, the stream data may evolve over time, and the corresponding data mining or query results may also change over time. Thus, the results of a query over a more recent window may be quite different from the results of a query over a more distant window. Similarly, the entire history of the data stream may not relevant for use in a repetitive data mining application such as classification. Recently, the reservoir sampling algorithm was adapted to sample from a moving window over data streams [8]. This is useful for data streams, since only a small amount of recent history is more relevant that the entire data stream. However, this can sometimes be an extreme solution, since one may desire to sample from varying lengths of the stream history. While recent queries may be more frequent, it is also not possible to completely disregard queries over more distant horizons in the data stream. A method in [4] designs methods for *biased reservoir sampling*, which uses a bias function to regulate the sampling from the stream. This bias function is quite effective since it regulates the sampling in a smooth way so that queries over recent horizons are more accurately resolved. While the design of a reservoir for arbitrary bias function is extremely difficult, it is shown in [4], that certain classes of bias functions (exponential bias functions) allow the use of a straightforward replacement algorithm. The advantage of a bias function is that it can smoothly regulate the sampling process so that acceptable accuracy is retained for more distant queries. The method in [4] can also be used in data mining applications so that the quality of the results do not degrade very quickly.

2.2 Concise Sampling

The effectiveness of the reservoir based sampling method can be improved further with the use of concise sampling. We note that the size of the reservoir is sometimes restricted by the available main memory. It is desirable to increase the sample size within the available main memory restrictions. For this purpose, the technique of concise sampling is quite effective.

The method of concise sampling exploits the fact that the number of *distinct* values of an attribute is often significantly smaller than the size of the data stream. This technique is most applicable while performing univariate sampling along a single dimension. For the case of multi-dimensional sampling, the simple reservoir based method discussed above is more appropriate. The repeated occurrence of the same value can be exploited in order to increase the sample size beyond the relevant space restrictions. We note that when the number of distinct values in the stream is smaller than the main memory limitations, the entire stream can be maintained in main memory, and therefore sampling may not even be necessary. For current desktop systems in which the memory sizes may be of the order of several gigabytes, very large sample sizes can be main memory resident, as long as the number of distinct values does not exceed the memory constraints. On the other hand, for more challenging streams with an unusually large number of distinct values, we can use the following approach.

The sample is maintained as a set S of <value, count> pairs. For those pairs in which the value of count is one, we do not maintain the count explicitly, but we maintain the value as a *singleton*. The number of elements in this representation is referred to as the footprint and is bounded above by n. We note that the footprint size is always smaller than or equal to than the true sample size. If the count of any distinct element is larger than 2, then the footprint size is strictly smaller than the sample size. We use a *threshold parameter* τ which defines the probability of successive sampling from the stream. The value of τ is initialized to be 1. As the points in the stream arrive, we add them to the current sample with probability $1/\tau$. We note that if the corresponding value-count pair is already included in the set S, then we only need to increment the count by 1. Therefore, the footprint size does not increase. On the other hand, if the value of the current point is distinct from all the values encountered so far, or it exists as a singleton then the foot print increases by 1. This is because either a singleton needs to be added, or a singleton gets converted to a value-count pair with a count of 2. The increase in footprint size may potentially require the removal of an element from sample S in order to make room for the new insertion. When this situation arises, we pick a new (higher) value of the threshold τ', and apply this threshold to the footprint in repeated passes. In each pass, we reduce the count of a value with probability τ/τ', until at least one value-count pair reverts to a singleton or a singleton is removed. Subsequent

Granularity (Order k)	**Averages** Φ values	**DWT Coefficients** ψ values
$k = 4$	(8, 6, 2, 3, 4, 6, 6, 5)	-
$k = 3$	(7, 2.5, 5, 5.5)	(1, -0.5,-1, 0.5)
$k = 2$	(4.75, 5.25)	(2.25, -0.25)
$k = 1$	(5)	(-0.25)

Table 9.1. An Example of Wavelet Coefficient Computation

points from the stream are sampled with probability $1/\tau'$. As in the previous case, the probability of sampling reduces with stream progression, though we have much more flexibility in picking the threshold parameters in this case. More details on the approach may be found in [41].

One of the interesting characteristics of this approach is that the sample S continues to remain an unbiased representative of the data stream irrespective of the choice of τ. In practice, τ' may be chosen to be about 10% larger than the value of τ. The choice of different values of τ provides different tradeoffs between the average (true) sample size and the computational requirements of reducing the footprint size. In general, the approach turns out to be quite robust across wide ranges of the parameter τ.

3. Wavelets

Wavelets [66] are a well known technique which is often used in databases for hierarchical data decomposition and summarization. A discussion of applications of wavelets may be found in [10, 66, 89]. In this chapter, we will discuss the particular case of the *Haar Wavelet*. This technique is particularly simple to implement, and is widely used in the literature for hierarchical decomposition and summarization. The basic idea in the wavelet technique is to create a decomposition of the data characteristics into a set of wavelet functions and basis functions. The property of the wavelet method is that the higher order coefficients of the decomposition illustrate the broad trends in the data, whereas the more localized trends are captured by the lower order coefficients.

We assume for ease in description that the length q of the series is a power of 2. This is without loss of generality, because it is always possible to decompose a series into segments, each of which has a length that is a power of two. The Haar Wavelet decomposition defines 2^{k-1} coefficients of order k. Each of these 2^{k-1} coefficients corresponds to a contiguous portion of the time series of length $q/2^{k-1}$. The ith of these 2^{k-1} coefficients corresponds to the segment in the series starting from position $(i-1) \cdot q/2^{k-1} + 1$ to position $i * q/2^{k-1}$. Let us denote this coefficient by ψ_k^i and the corresponding time series segment by S_k^i. At the same time, let us define the average value of the first half of the S_k^i by

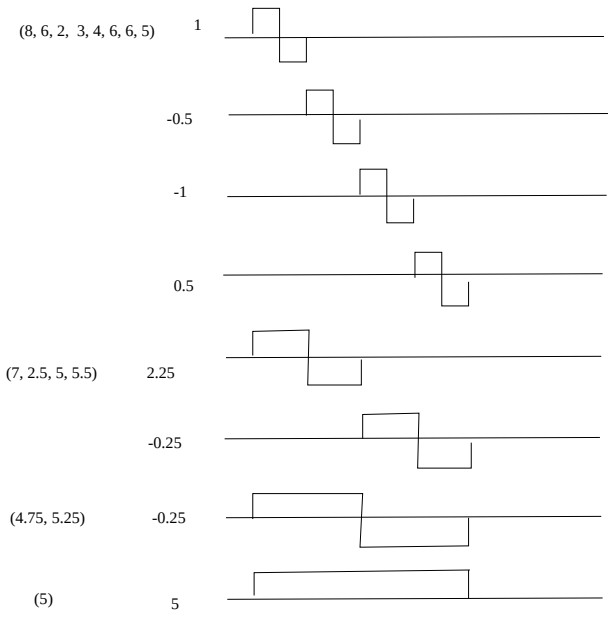

Figure 9.1. Illustration of the Wavelet Decomposition

a_k^i and the second half by b_k^i. Then, the value of ψ_k^i is given by $(a_k^i - b_k^i)/2$. More formally, if Φ_k^i denote the average value of the S_k^i, then the value of ψ_k^i can be defined recursively as follows:

$$\psi_k^i = (\Phi_{k+1}^{2 \cdot i-1} - \Phi_{k+1}^{2 \cdot i})/2 \tag{9.6}$$

The set of Haar coefficients is defined by the Ψ_k^i coefficients of order 1 to $\log_2(q)$. In addition, the global average Φ_1^1 is required for the purpose of perfect reconstruction. We note that the coefficients of different order provide an understanding of the major trends in the data at a particular level of granularity. For example, the coefficient ψ_k^i is half the quantity by which the first half of the segment S_k^i is larger than the second half of the same segment. Since larger values of k correspond to geometrically reducing segment sizes, one can obtain an understanding of the basic trends at different levels of granularity. We note that this definition of the Haar wavelet makes it very easy to compute by a sequence of averaging and differencing operations. In Table 9.1, we have illustrated how the wavelet coefficients are computed for the case of the sequence $(8, 6, 2, 3, 4, 6, 6, 5)$. This decomposition is illustrated in graphical form in Figure 9.1. We also note that each value can be represented as a sum of $\log_2(8) = 3$ linear decomposition components. In general, the entire decomposition may be represented as a tree of depth 3, which represents the

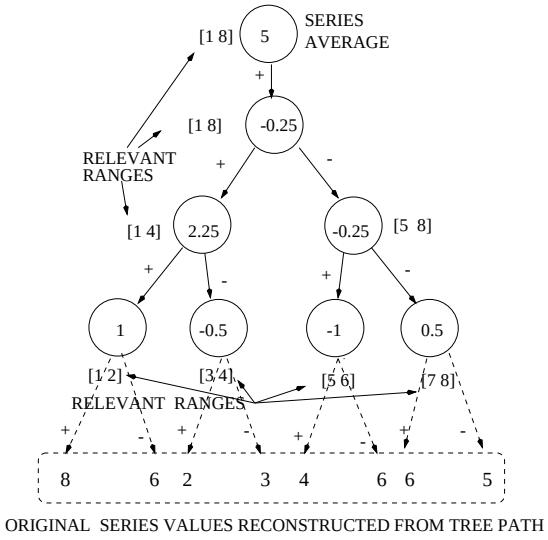

Figure 9.2. The Error Tree from the Wavelet Decomposition

hierarchical decomposition of the entire series. This is also referred to as the *error tree*, and was introduced in [73]. In Figure 9.2, we have illustrated the error tree for the wavelet decomposition illustrated in Table 9.1. The nodes in the tree contain the values of the wavelet coefficients, except for a special *super-root* node which contains the series average. This super-root node is not necessary if we are only considering the relative values in the series, or the series values have been normalized so that the average is already zero. We further note that the number of wavelet coefficients in this series is 8, which is also the length of the original series. The original series has been replicated just below the error-tree in Figure 9.2, and it can be reconstructed by adding or subtracting the values in the nodes along the path leading to that value. We note that each coefficient in a node should be added, if we use the left branch below it to reach to the series values. Otherwise, it should be subtracted. This natural decomposition means that an entire contiguous range along the series can be reconstructed by using only the portion of the error-tree which is relevant to it. Furthermore, we only need to retain those coefficients whose values are significantly large, and therefore affect the values of the underlying series. In general, we would like to minimize the reconstruction error by retaining only a fixed number of coefficients, as defined by the space constraints.

We further note that the coefficients represented in Figure 9.1 are un-normalized. For a time series T, let $\overline{W_1} \ldots \overline{W_t}$ be the corresponding basis vectors of length t. In Figure 9.1, each component of these basis vectors is 0, +1, or -1. The list

of basis vectors in Figure 9.1 (in the same order as the corresponding wavelets illustrated) are as follows:

(1 -1 0 0 0 0 0 0)
(0 0 1 -1 0 0 0 0)
(0 0 0 0 1 -1 0 0)
(0 0 0 0 0 0 1 -1)
(1 1 -1 -1 0 0 0 0)
(0 0 0 0 1 1 -1 -1)
(1 1 1 1 -1 -1 -1 -1)

The most detailed coefficients have only one +1 and one -1, whereas the most coarse coefficient has $t/2$ +1 and -1 entries. Thus, in this case, we need $2^3 - 1 = 7$ wavelet vectors. In addition, the vector (11111111) is needed to represent the special coefficient which corresponds to the series average. Then, if $a_1 \ldots a_t$ be the wavelet coefficients for the wavelet vectors $\overline{W_1} \ldots \overline{W_t}$, the time series T can be represented as follows:

$$T = \sum_{i=1}^{t} a_i \cdot \overline{W_i} \tag{9.7}$$

$$= \sum_{i=1}^{t} (a_i \cdot |\overline{W_i}|) \cdot \frac{\overline{W_i}}{|\overline{W_i}|} \tag{9.8}$$

While a_i is the un-normalized value from Figure 9.1, the values $a_i \cdot |\overline{W_i}|$ represent normalized coefficients. We note that the values of $|\overline{W_i}|$ are different for coefficients of different orders, and may be equal to either $\sqrt{2}$, $\sqrt{4}$ or $\sqrt{8}$ in this particular example. For example, in the case of Figure 9.1, the broadest level unnormalized coefficient is -0.25, whereas the corresponding normalized value is $-0.25 \cdot \sqrt{8}$. After normalization, the basis vectors $\overline{W_1} \ldots \overline{W_t}$ are orthonormal, and therefore, the sum of the squares of the corresponding (normalized) coefficients is equal to the energy in the time series T. Since the normalized coefficients provide a new coordinate representation after axis rotation, euclidian distances between time series are preserved in this new representation.

The total number of coefficients is equal to the length of the data stream. Therefore, for very large time series or data streams, the number of coefficients is also large. This makes it impractical to retain the entire decomposition throughout the computation. The wavelet decomposition method provides a natural method for dimensionality reduction, by retaining only the coefficients with large absolute values. All other coefficients are implicitly approximated to zero. This makes it possible to approximately represent the series with a small number of coefficients. The idea is to retain only a pre-defined number of coefficients from the decomposition, so that the error of the reduced representation is minimized. Wavelets are used extensively for efficient and approximate

query processing of different kinds of data [11, 93]. They are particularly useful for range queries, since contiguous ranges can easily be reconstructed with a small number of wavelet coefficients. The efficiency of the query processing arises from the reduced representation of the data. At the same time, since only the small coefficients are discarded the results are quite accurate.

A key issue for the accuracy of the query processing is the choice of coefficients which should be retained. While it may be tempting to choose only the coefficients with large absolute values, this is not always the best choice, since a more judicious choice of coefficients can lead to minimizing specific error criteria. Two such metrics are the minimization of the mean square error or the maximum error metric. The mean square error minimizes the L_2 error in approximation of the wavelet coefficients, whereas maximum error metrics minimize the maximum error of any coefficient. Another related metric is the relative maximum error which normalizes the maximum error with the absolute coefficient value.

It has been shown in [89] that the choice of largest B (normalized) coefficients minimizes the mean square error criterion. This should also be evident from the fact that the normalized coefficients render an orthonormal decomposition, as a result of which the energy in the series is equal to the sum of the squares of the coefficients. However, the use of the mean square error metric is not without its disadvantages. A key disadvantage is that a global optimization criterion implies that the local behavior of the approximation is ignored. Therefore, the approximation arising from reconstruction can be arbitrarily poor for certain regions of the series. This is especially relevant in many streaming applications in which the queries are performed only over recent time windows. In many cases, the maximum error metric provides much more robust guarantees. In such cases, the errors are spread out over the different coefficients more evenly. As a result, the worst-case behavior of the approximation over different queries is much more robust.

Two such methods for minimization of maximum error metrics are discussed in [38, 39]. The method in [38] is probabilistic, but its application of probabilistic expectation is questionable according to [53]. One feature of the method in [38] is that the space is bounded only in expectation, and the variance in space usage is large. The technique in [39] is deterministic and uses dynamic programming in order to optimize the maximum error metric. The key idea in [39] is to define a recursion over the nodes of the tree in top down fashion. For a given internal node, we compute the least maximum error over the two cases of either keeping or not keeping a wavelet coefficient of this node. In each case, we need to recursively compute the maximum error for its two children over all possible space allocations among two children nodes. While the method is quite elegant, it is computationally intensive, and it is therefore not suitable for the data stream case. We also note that the coefficient is defined according to

the wavelet coefficient definition i.e. half the difference between the left hand and right hand side of the time series. While this choice of coefficient is optimal for the L_2 metric, this is not the case for maximum or arbitrary L_p error metrics.

Another important topic in wavelet decomposition is that of the use of multiple measures associated with the time series. The problem of multiple measures refers to the fact that many quantities may simultaneously be tracked in a given time series. For example, in a sensor application, one may simultaneously track many variables such as temperature, pressure and other parameters at each time instant. We would like to perform the wavelet decomposition over multiple measures simultaneously. The most natural technique [89] is to perform the decomposition along the different measures separately and pick the largest coefficients for each measure of the decomposition. This can be inefficient, since a coordinate needs to be associated with each separately stored coefficient and it may need to be stored multiple times. It would be more efficient to amortize the storage of a coordinate across multiple measures. The trade-off is that while a given coordinate may be the most effective representation for a particular measure, it may not simultaneously be the most effective representation across all measures. In [25], it has been proposed to use an extended wavelet representation which simultaneously tracks multi-measure coefficients of the wavelet representation. The idea in this technique is use a bitmap for each coefficient set to determine which dimensions are retained, and store all coefficients for this coordinate. The technique has been shown to significantly outperform the methods discussed in [89].

3.1 Recent Research on Wavelet Decomposition in Data Streams

The one-pass requirement of data streams makes the problem of wavelet decomposition somewhat more challenging. However, the case of optimizing the mean square error criterion is relatively simple, since a choice of the largest coefficients can preserve the effectiveness of the decomposition. Therefore, we only need to dynamically construct the wavelet decomposition, and keep track of the largest B coefficients encountered so far.

As discussed in [65], these methods can have a number of disadvantages in many situations, since many parts of the time series may be approximated very poorly. The method in [39] can effectively perform the wavelet decomposition with maximum error metrics. However, since the method uses dynamic programming, it is computationally intensive, it is quadratic in the length of the series. Therefore, it cannot be used effectively for the case of data streams, which require a one-pass methodology in linear time. in [51], it has been shown that all weighted L_m measures can be solved in a space-efficient manner using only $O(n)$ space. In [65], methods have been proposed for one-pass wavelet

synopses with the maximum error metric. It has been shown in [65], that by using a number of intuitive thresholding techniques, it is possible to approximate the effectiveness of the technique discussed in [39]. A set of independent results obtained in [55] discuss how to minimize non-euclidean and relative error with the use of wavelet synopses. This includes metrics such as the L_p error or the relative error. Both the works of [65] and [55] were obtained independently and at a similar time. While the method in [65] is more deeply focussed on the use of maximum error metrics, the work in [55] also provides some worst case bounds on the quality of the approximation. The method of [65] depends on experimental results to illustrate the quality of the approximation. Another interesting point made in [55] is that most wavelet approximation methods solve a restricted version of the problem in which the wavelet coefficient for the basis is defined to be half the difference between the left hand and right hand side of the basis vectors. Thus, the problem is only one of picking the best B coefficients out of these pre-defined set of coefficients. While this is an intuitive method for computation of the wavelet coefficient, and is optimal for the case of the Euclidean error, it is not necessarily optimal for the case of the L_m-metric. For example, consider the time series vector $(1, 4, 5, 6)$. In this case, the wavelet transform is $(4, -1.5, -1.5, -0.5)$. Thus, for $B = 1$, the optimal coefficient picked is $(4, 0, 0, 0)$ for any L_m-metric. However, for the case of L_∞-metric, the optimal solution should be $(3.5, 0, 0, 0)$, since 3.5 represents the average between the minimum and maximum value. Clearly, any scheme which restricts itself only to wavelet coefficients defined in a particular way will not even consider this solution [55]. Almost all methods for non-euclidean wavelet computation tend to use this approach, possibly as a legacy from the Haar method of wavelet decomposition. This restriction has been removed in [55] and proposes a method for determining the optimal *synopsis coefficients* for the case of the weighted L_m metric. We distinguish between synopsis coefficients and wavelet coefficients, since the latter are defined by the simple subtractive methodology of the Haar decomposition. A related method was also proposed by Matias and Urieli [75] which discusses a near linear time optimal algorithm for the weighted L_m-error. This method is offline, and chooses a basis vector which depends upon the weights.

An interesting extension of the wavelet decomposition method is one in which *multiple measures* are associated with the time series. A natural solution is to treat each measure separately, and store the wavelet decomposition. However, this can be wasteful, since a coordinate needs to be stored with each coefficient, and we can amortize this storage by storing the same coordinate across multiple measures. A technique in [25] proposes the concept of *extended wavelets* in order to amortize the coordinate storage across multiple measures. In this representation, one or more coefficients are stored with each coordinate. Clearly, it can be tricky to determine which coordinates to store,

since different coordinates will render larger coefficients across different measures. The technique in [25] uses a dynamic programming method to determine the optimal extended wavelet decomposition. However, this method is not time and space efficient. A method in [52] provides a fast algorithm whose space requirement is linear in the size of the synopsis and logarithmic in the size of the data stream.

Another important point to be noted is that the choice of the best wavelet decomposition is not necessarily pre-defined, but it depends upon the particular workload on which the wavelet decomposition is applied. Some interesting papers in this direction [77, 75] design methods for workload aware wavelet synopses of data streams. While this line of work has not been extensively researched, we believe that it is likely to be fruitful in many data stream scenarios.

4. Sketches

The idea of sketches is essentially an extension of the random projection technique [64] to the time series domain. The idea of using this technique for determining representative trends in the time series domain was first observed in [61]. In the method of random projection, we can reduce a data point of dimensionality d to an axis system of dimensionality k by picking k random vectors of dimensionality d and calculating the dot product of the data point with each of these random vectors. Each component of the k random vectors is drawn from the normal distribution with zero mean and unit variance. In addition, the random vector is normalized to one unit in magnitude. It has been shown in [64] that proportional L_2 distances between the data points are approximately preserved using this transformation. The accuracy bounds of the distance values are dependent on the value of k. The larger the chosen value of k, the greater the accuracy and vice-versa.

This general principle can be easily extended to the time series domain, by recognizing the fact that the length of a time series may be treated as its dimensionality, and correspondingly we need to compute a random vector of length equal to the time series, and use it for the purpose of sketch computation. If desired, the same computation can be performed over a sliding window of a given length by choosing a random vector of appropriate size. As proposed in [61], the following approximation bounds are preserved:

LEMMA 9.1 *Let L be a set of vectors of length l, for fixed $\epsilon < 1/2$, and $k = 9 \cdot log|L|/\epsilon^2$. Consider a pair of vectors $\overline{u}, \overline{w}$ in L, such that the corresponding sketches are denoted by $S(\overline{u})$ and $S(\overline{w})$ respectively. Then, we have:*

$$(1-\epsilon) \cdot ||\overline{u} - \overline{w}||^2 \leq ||S(\overline{u}) - S(\overline{w})|| \leq (1+\epsilon) \cdot ||\overline{u} - \overline{w}||^2 \tag{9.9}$$

with probability $1/2$. Here $||U - V||^2$ is the L_2 distance between two vectors U and V.

The generalization to time series is fairly straightforward, and the work in [61] makes two primary contributions in extending the sketch methodology to finding time series trends.

4.1 Fixed Window Sketches for Massive Time Series

In this case, we wish to determine sliding window sketches with a fixed window length l. For each window of length l, we need to perform $l \cdot k$ operations for a sketch of size k. Since there are $O(n - l)$ sliding windows, this will require $O(n \cdot l \cdot k)$ computations. When l is large, and is of the same order of magnitude as the time series, the computation may be quadratic in the size of the series. This can be prohibitive for very large time series, as is usually the case with data streams. The key observation in [61], is that all such sketches can be viewed as the problem of computing the polynomial convolution of the random vector of appropriate length with the time series. Since the problem of polynomial convolution can be computed efficiently using fast fourier transform, this also means that the sketches may be computed efficiently. The problem of polynomial convolution is defined as follows:

DEFINITION 9.2 *Given two vectors $A[1 \ldots a]$ and $B[1 \ldots b]$, $a \geq b$, their convolution is the vector $C[1 \ldots a+b]$ where $C[k] = \sum_{i=1}^{b} A[k-i] \cdot B[i]$ for $k \in [2, a+b]$, with any out of range references assumed to be zero.*

The key point here is that the above polynomial convolution can be computed using FFT, in $O(a \cdot \log(b))$ operations rather than $O(a \cdot b)$ operations. This effectively means the following:

LEMMA 9.3 *Sketches of all subvectors of length l can be computed in time $O(n \cdot k \cdot log(l))$ using polynomial convolution.*

4.2 Variable Window Sketches of Massive Time Series

The method in the previous subsection discussed the problem of sketch computation for a fixed window length. The more general version of the problem is one in which we wish to compute the sketch for any subvector between length l and u. In the worst-case this comprises $O(n^2)$ subvectors, most of which will have length $O(n)$. Therefore, the entire algorithm may require $O(n^3)$ operations, which can be prohibitive for massive time series streams.

The key idea in [61] is to store a *pool* of sketches. The size of this pool is significantly smaller than the entire set of sketches needed. However, it is carefully chosen so that the sketch of any sub-vector in the original vector can be computed in $O(1)$ time fairly accurately. In fact, it can be shown that the approximate sketches computed using this approach satisfy a slightly relaxed version of Lemma 9.1. We refer details to [61].

4.3 Sketches and their applications in Data Streams

In the previous sections we discussed the application of sketches to the problem of massive time series. Some of the methods such as fixed window sketch computation are inherently offline. This does not suffice in many scenarios in which it is desirable to continuously compute the sketch over the data stream. Furthermore, in many cases, it is desirable to efficiently use this sketch in order to work with a variety of applications such as query estimation. In this subsection, we will discuss the applications of sketches in the data stream scenario. Our earlier discussion corresponds to a sketch of the *time series* itself, and entails the storage of the random vector required for sketch generation. While such a technique can be used effectively for massive time series, it cannot always be used for time series data streams.

However, in certain other applications, it may be desirable to track the *frequencies* of the distinct values in the data stream. In this case, if $(u_1 \ldots u_N)$ be the frequencies of N distinct values in the data stream, then the sketch is defined by the dot product of the vector $(u_1 \ldots u_N)$ with a random vector of size N. As in the previous case, the number of distinct items N may be large, and therefore the size of the corresponding random vector will also be large. A natural solution is to pre-generate a set of k random vectors, and whenever the ith item is received, we add r_i^j to the jth sketch component. Therefore, the k random vectors may need to be pre-stored in order to perform the computation. However, the explicit storage of the random vector will defeat the purpose of the sketch computation, because of the high space complexity.

The key here is to store the random vectors implicitly in the form of a seed, which can be used to dynamically generate the vector. The key idea discussed in [6] is that it is possible to generate the random vectors from a seed of size $O(\log(N))$ provided that one is willing to work with the restriction that the values of $r_i^j \in \{-1, +1\}$ are only 4-wise independent. We note that having a seed of small size is critical in terms of the space-efficiency of the method. Furthermore, it has been shown in [6] that the theoretical results only require 4-wise independence. In [44], it has also been shown how to use Reed-Muller codes in order to generate 7-wise independent random numbers. These method suffices for the purpose of wavelet decomposition of the frequency distribution of different items.

Some key properties of the pseudo-random number generation approach and the sketch representation are as follows:

- A given component r_i^j can be generated in poly-logarithmic time from the seed.
- The dot-product of two vectors can be approximately computed using only their sketch representations. This follows from the fact that the

dot product of two vectors is closely related to the Euclidean distance, a quantity easily approximated by the random projection approach [64]. Specifically, if $\overline{U}$ and $\overline{V}$ be two (normalized) vectors, then the euclidean distance and dot product are related as follows:

$$||\overline{U} - \overline{V}||^2 = ||\overline{U}||^2 + ||\overline{V}||^2 - 2 \cdot \overline{U} \cdot \overline{V} \tag{9.10}$$

$$\tag{9.11}$$

This relationship can be used to establish bounds on the quality of the dot product approximation of the sketch vectors. We refer to [44] for details of the proof.

The first property ensures that the sketch components can be updated and maintained efficiently. Whenever the ith value is received, we only need to add r_i^j to the jth component of the sketch vector. Since the quantity r_i^j can be efficiently computed, it follows that the update operations can be performed efficiently as well. In the event that the data stream also incorporates frequency counts with the arriving items (item i is associated with frequency count $f(i)$), then we simply need to add $f(i) \cdot r_i^j$ to the jth sketch component. We note that the efficient and accurate computation of the dot product of a given time series with the random vector is a key primitive which can be used to compute many properties such as the wavelet decomposition. This is because each wavelet coefficient can be computed as the dot product of the wavelet basis with the corresponding time series data stream; an approximation may be determined by using only their sketches. The key issue here is that we also need the sketch representation of the wavelet basis vectors, each of which may take $O(N)$ time in the worst case. In general, this can be time consuming; however the work in [44] shows how to do this in poly-logarithmic time for the special case in which the vectors are Haar-basis vectors. Once the coefficients have been computed, we only need to retain the B coefficients with the highest energy.

We note that one property of the results in [44] is that it uses the sketch representation of the frequency distribution of the *original stream* in order to derive the wavelet coefficients. A recent result in [16] works *directly* with the sketch representation of the wavelet coefficients rather than the sketch representation of the original data stream. Another advantage of the work in [16] is that the query times are much more efficient, and the work extends to the multi-dimensional domain. We note that while the wavelet representation in [44] is space efficient, the entire synopsis structure may need to be touched for updates and every wavelet coefficient must be touched in order to find the best B coefficients. The technique in [16] reduces the time and space efficiency for both updates and queries.

The method of sketches can be effectively used for second moment and join estimation. First, we discuss the problem of second moment estimation [6] and

illustrate how it can be used for the problem of estimating the size of self joins. Consider a set of n quantitative values $U = (u_1 \ldots u_N)$. We would like to estimate the second moment $|U|^2$. Then, as before generate the random vectors $\overline{r^1} \ldots r^k$, (each of size N), and compute the dot product of these random vectors with U to create the sketch components denoted by $S_1 \ldots S_k$. Then, it can be shown that the expected value of S_i^2 is equal to the second moment. In fact, the approximation can be bounded with high probability.

LEMMA 9.4 *By selecting the median of $O(log(1/\delta))$ averages of $O(1/\epsilon^2)$ copies of S_i^2, it is possible to guarantee the accuracy of the sketch based approximation to within $1 \pm \epsilon$ with probability at least $1 - \delta$.*

In order to prove the above result, the first step is to show that the expected value of S_i^2 is equal to the second moment, and the variance of the variable S_i^2 is at most twice the square of the expected value. The orthogonality of the random projection vectors can be used to show the first result and the 4-wise independence of the values of r_i^j can be used to show the second. The relationship between the expected values and variance imply that the Chebychev inequality can be used to prove that the average of $O(1/\epsilon^2)$ copies provides the ϵ bound with a constant probability which is at least $7/8$. This constant probability can be tightened to at least $1 - \delta$ (for any small value of δ) with the use of the median of $O(\log(1/\delta))$ independent copies of these averages. This is because the median would lie outside the ϵ-bound only if more than $\log(1/\delta)/2$ copies (minimum required number of copies) lie outside the ϵ bound. However, the expected number of copies which lie outside the ϵ-bound is only $\log(1/\delta)/8$, which is less than above-mentioned required number of copies by $3 \cdot \log(1/\delta)/8$. The Chernoff tail bounds can then be applied on the random variable representing the number of copies lying outside the ϵ-bound. This can be used to show that the probability of more than half the $\log(1/\delta)$ copies lying outside the ϵ-bound is at most δ. Details of the proof can be found in [6].

We note that the second moment immediately provides an estimation for self-joins. If u_i be the number of items corresponding to the ith value, then the second moment estimation is exactly the size of the self-join. We further note that the dot product function is not the only one which can be estimated from the sketch. In general, many functions such as the dot product, the L_2 distance, or the maximum frequency items can be robustly estimated from the sketch. This is essentially because the sketch simply projects the time series onto a new set of (expected) orthogonal vectors. Therefore many rotational invariant properties such as the L_2 distance, dot product, or second moment are approximately preserved by the sketch.

A number of interesting techniques have been discussed in [5, 26, 27] in order to perform the estimation more effectively over general joins and multi-joins. Consider the multi-join problem on relations $R1$, $R2$, $R3$, in which we

wish to join attribute A of $R1$ with attribute B of R2, and attribute C of $R2$ with attribute D of $R3$. Let us assume that the join attribute on $R1$ with $R2$ has N distinct values, and the join attribute of $R2$ with $R3$ has M distinct values. Let $f(i)$ be the number of tuples in $R1$ with value i for attribute A. Let $g(i, j)$ be the number of tuples in $R2$ with values i and j for attributes B and C respectively. Let $h(j)$ be the number of tuples in $R3$ with value j for join attribute C. Then, the total estimated join size J is given by the following:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{M} f(i) \cdot g(i, j) \cdot h(j) \tag{9.12}$$

In order to estimate the join size, we create *two independently* generated families of random vectors $\overline{r^1} \ldots \overline{r^k}$ and $\overline{s^1} \ldots \overline{s^k}$. We dynamically maintain the following quantities, as the stream points are received:

$$Z_1^j = \sum_{i=1}^{N} f(i) \cdot r_i^j \tag{9.13}$$

$$Z_2^j = \sum_{i=1}^{N} \sum_{k=1}^{M} g(i, k) \cdot r_k^j \cdot s_k^j \tag{9.14}$$

$$Z_3^j = \sum_{k=1}^{M} h(k) \cdot s_k^j \tag{9.15}$$

It can be shown [5], that the quantity $Z_1^j \cdot Z_2^j \cdot Z_3^j$ estimates the join size. We can use the multiple components of the sketch (different values of j) in order to improve the accuracy. It can be shown that the variance of this estimate is equal to the *product of the self-join sizes* for the three different relations. Since the tail bounds use the variance in order to provide quality estimates, a large value of the variance can reduce the effectiveness of such bounds. This is particularly a problem if the composite join has a small size, whereas the product of the self-join sizes is very large. In such cases, the errors can be very large in relation to the size of the result itself. Furthermore, the product of self-join sizes increases with the number of joins. This degrades the results. We further note that the error bound results for sketch based methods are proved with the use of the Chebychev inequality, which depends upon a low ratio of the variance to result size. A high ratio of variance to result size makes this inequality ineffective, and therefore the derivation of worst-case bounds requires a greater number of sketch components.

An interesting observation in [26] is that of *sketch partitioning*. In this technique, we intelligently partition the join attribute domain-space and use it in order to compute separate sketches of each partition. The resulting join

estimate is computed as the sum over all partitions. The key observation here is that intelligent domain partitioning reduces the variance of the estimate, and is therefore more accurate for practical purposes. This method has also been discussed in more detail for the problem of multi-query processing [27].

Another interesting trick for improving join size estimation is that of *sketch skimming* [34]. The key insight is that the variance of the join estimation is highly affected by the most frequent components, which are typically small in number. A high variance is undesirable for accurate estimations. Therefore, we treat the frequent items in the stream specially, and can separately track them. A skimmed sketch can be constructed by subtracting out the sketch components of these frequent items. Finally, the join size can be estimated as a 4-wise addition of the join estimation across two pairs of partitions. It has been shown that this approach provides superior results because of the reduced variance of the estimations from the skimmed sketch.

4.4 Sketches with p-stable distributions

In our earlier sections, we did not discuss the effect of the distribution from which the random vectors are drawn. While the individual components of the random vector were drawn from the normal distribution, this is not the only possibility for sketch generation. In this section, we will discuss a special set of distributions for the random vectors which are referred to as p-stable distributions. A distribution $\mathcal{L}$ is said to be p-stable, if it satisfies the following property:

DEFINITION 9.5 *For any set of N i.i.d. random variables $X_1 \ldots X_N$ drawn from a p-stable distribution $\mathcal{L}$, and any set of real numbers $a_1 \ldots a_N$, the random variable $(\sum_{i=1}^{N} a_i \cdot X_i)/(\sum_{i=1}^{N} a_i^p)^{(1/p)}$ is drawn from $\mathcal{L}$.*

A classic example of the p-stable distribution is the normal distribution with $p = 2$. In general p-stable distributions can be defined for $p \in (0, 2]$.

The use of p-stable distributions has implications in the construction of sketches. Recall, that the ith sketch component is of the form $\sum_{i=1}^{N} u_j \cdot r_i^j$, where u_i is the frequency of the ith distinct value in the data stream. If each r_i^j is drawn from a p-stable distribution, then the above sum is also a (scaled) p-stable distribution, where the scale coefficient is given by $(\sum_{i=1}^{N} u_i^p)^{(1/p)}$. The ability to use the *exact distribution* of the sketch provides much stronger results than just the use of mean and variance of the sketch components. We note that the use of only mean and variance of the sketch components often restricts us to the use of generic tail bounds (such as the Chebychev inequality) which may not always be tight in practice. However, the knowledge of the sketch distribution can potentially provide very tight bounds on the behavior of each sketch component.

An immediate observation is that the scale coefficient $(\sum_{i=1}^{N} u_i^p)^{(1/p)}$ of each sketch component is simply the L_p-norm of the frequency distribution of the incoming items in the data stream. By using $O(\log(1/\delta)/\epsilon^2)$ independent sketch components, it is possible to approximate the L_p norm within ϵ with probability at least $1-\delta$. We further note that the use of the L_0 norm provides the number of distinct values in the data stream. It has been shown in [17] that by using $p \to 0$ (small values of p), it is possible to closely approximate the number of distinct values in the data stream.

Other Applications of Sketches. The method of sketches can be used for a variety of other applications. Some examples of such applications include the problem of *heavy hitters* [13, 18, 76, 21], a problem in which we determine the most frequent items over data streams. Other problems include those of finding significant network differences over data streams [19] and finding quantiles [46, 50] over data streams. Another interesting application is that of significant differences between data streams [32, 33], which has applications in numerous change detection scenarios. Another recent application to sketches has been to XML and tree-structured data [82, 83, 87]. In many cases, these synopses can be used for efficient resolution of the structured queries which are specified in the XQuery pattern-specification language.

Recently sketch based methods have found considerable applications to efficient communication of signals in sensor networks. Since sensors are battery constrained, it is critical to reduce the communication costs of the transmission. The space efficiency of the sketch computation approach implies that it can also be used in the sensor network domain in order to minimize the communication costs over different processors. In [22, 67, 50], it has been shown how to extend the sketch method to distributed query tracking in data streams. A particularly interesting method is the technique in [22] which reduces the communication costs further by using sketch skimming techniques [34], in order to reduce communication costs further. The key idea is to use models to estimate the future behavior of the sketch, and make changes to the sketch only when there are significant changes to the underlying model.

4.5 The Count-Min Sketch

One interesting variation of the sketching method for data streams is the *count-min sketch*, which uses a hash-based sketch of the stream. The broad ideas in the count-min sketch were first proposed in [13, 29, 30]. Subsequently, the method was enhanced with pairwise-independent hash functions, formalized, and extensively analyzed for a variety of applications in [20].

In the count-min sketch, we use $\lceil \ln(1/\delta) \rceil$ pairwise independent hash functions, each of which map on to uniformly random integers in the range $[0, e/\epsilon]$,

where e is the base of the natural logarithm. Thus, we maintain a total of $\lceil \ln(1/\delta) \rceil$ hash tables, and there are a total of $O(\ln(1/\delta)/\epsilon)$ hash cells. This *apparently* provides a better space complexity than the $O(\ln(1/\delta)/\epsilon^2)$ bound of AMS sketches in [6]. We will discuss more on this point later.

We apply each hash function to any incoming element in the data stream, and add the count of the element to each of the corresponding $\lceil \ln(1/\delta) \rceil$ positions in the different hash tables. We note that because of collisions, the hash table counts will not exactly correspond to the count of any element in the incoming data stream. When incoming frequency counts are non-negative, the hash table counts will over-estimate the true count, whereas when the incoming frequency counts are either positive or negative (deletions), the hash table count could be either an over-estimation or an under-estimation. In either case, the use of the median count of the hash position of a given element among the $O(\ln(1/\delta))$ counts provided by the different hash functions provides a estimate which is within a $3 \cdot \epsilon$ factor of the L_1-norm of element counts with probability at least $1 - \delta^{1/4}$ [20]. In other words, if the frequencies of the N different items are $f_1 \ldots f_N$, then the estimated frequency of the item i lie between $f_i - 3 \cdot \epsilon \cdot \sum_{i=1}^{N} |f_i|$ and $f_i + 3 \cdot \epsilon \cdot \sum_{i=1}^{N} |f_i|$ with probability at least $1 - \delta^{1/4}$. The proof of this result relies on the fact that the expected inaccuracy of a given entry j is at most $\epsilon \cdot \sum_{i=1}^{N} |f_i|/e$, if the hash function is sufficiently uniform. This is because we expect the count of other (incorrect) entries which map onto the position of j to be $\sum_{i \in [1,N], i \neq j} f_i \cdot \epsilon/e$ for a sufficiently uniform hash function with $\lceil e/\epsilon \rceil$ entries. This is at most equal to $\epsilon \cdot \sum_{i=1}^{N} |f_i|/e$. By the Markov inequality, the probability of this number exceeding $3 \cdot \epsilon \cdot \sum_{i=1}^{N} |f_i|$ is less than $1/(3 \cdot e) < 1/8$. By using the earlier Chernoff bound trick (as in AMS sketches) in conjunction with the median selection operation, we get the desired result.

In the case of non-negative counts, the *minimum count* of any of the $\ln(1/\delta)$ possibilities provides a tighter ϵ-bound (of the L_1-norm) with probability at least $1 - \delta$. In this case, the estimated frequency of item i lies between f_i and $f_i + \epsilon \cdot \sum_{i=1}^{N} f_i$ with probability at least $1 - \delta$. As in the previous case, the expected inaccuracy is $\epsilon \cdot \sum_{i=1}^{N} f_i/e$. This is less than the maximum bound by a factor of e. By applying the Markov inequality, it is clear that the probability that the bound is violated for a given entry is $1/e$. Therefore, the probability that it is violated by all $\log(1/\delta)$ entries is at most $(1/e)^{\log(1/\delta)} = \delta$.

For the case of non-negative vectors, the dot product can be estimated by computing the dot product on the corresponding entries in the hash table. Each of the $\lceil \ln(1/\delta) \rceil$ such dot products is an over estimate, and the minimum of these provides an ϵ bound with probability at least $1 - \delta$. The dot product result immediately provides bounds for join size estimation. Details of extending the method to other applications such as heavy hitters and quantiles may be found

in [20]. In many of these methods, the time and space complexity is bounded above by $O(\ln(1/\delta)/\epsilon)$, which is again apparently superior to the AMS sketch.

As noted in [20], the ϵ-bound in the count-min sketch cannot be directly compared with that of the AMS sketch. This is because the AMS sketch provides the ϵ-bound as a function of the L_2-norm, whereas the method in [20] provides the ϵ-bound only in terms of the L_1-norm. The L_1-norm can be *quadratically* larger (than the L_2-norm) in the most challenging case of non-skewed distributions, and the ratio between the two may be as large as $\sqrt{N}$. Therefore, the *equivalent* value of ϵ in the count-min sketch can be smaller than that in the AMS sketch by a factor of $\sqrt{N}$. Since N is typically large, and is in fact the motivation of the sketch-based approach, the *worst-case* time and space complexity of a truly equivalent count-min sketch may be *significantly* larger for practical values of ϵ. While this observation has been briefly mentioned in [20], there seems to be some confusion on this point in the current literature. This is because of the overloaded use of the parameter ϵ, which has different meaning for the AMS and count-min sketches. For the skewed case (which is quite common), the ratio of the L_1-norm to the L_2-norm reduces. However, since this case is less challenging, the general methods no longer remain relevant, and a number of other specialized methods (eg. sketch skimming [34]) exist in order to improve the experimental and worst-case effectiveness of both kinds of sketches. It would be interesting to experimentally compare the count-min and AMS methods to find out which is superior in different kinds of skewed and non-skewed cases. Some recent results [91] seem to suggest that the count-min sketch is experimentally superior to the AMS sketch in terms of maintaining counts of elements. On the other hand, the AMS sketch seems to be superior in terms of estimating aggregate functions such as the L_2-norm. Thus, the count-min sketch does seem to have a number of practical advantages in many scenarios.

4.6 Related Counting Methods: Hash Functions for Determining Distinct Elements

The method of sketches is a probabilistic counting method whereby a randomized function is applied to the data stream in order to perform the counting in a space-efficient way. While sketches are a good method to determine *large aggregate signals*, they are not very useful for counting infrequently occurring items in the stream. For example, problems such as the determination of the number of distinct elements cannot be performed with sketches. For this purpose, hash functions turn out to be a useful choice.

Consider a hash function that renders a mapping from a given word to an integer in the range $[0, 2^L - 1]$. Therefore, the binary representation of that integer will have length L. The position (least significant and rightmost bit is counted as 0) of the rightmost 1-bit of the binary representation of that integer

is tracked, and the largest such value is retained. This value is logarithmically related to the number of distinct elements [31] in the stream.

The intuition behind this result is quite simple. For a sufficiently uniformly distributed hash function, the probability of the ith bit on the right taking on the first 1-value is simply equal to 2^{-i-1}. Therefore, for N distinct elements, the expected number of records taking on the ith bit as the first 1-value is $2^{-i-1} \cdot N$. Therefore, when i is picked larger than $\log(N)$, the expected number of such bitstrings falls off exponentially less than 1. It has been rigorously shown [31] that the expected position of the rightmost bit $E[R]$ is logarithmically related to the number of distinct elements as follows:

$$E[R] = \log_2(\phi N), \quad \phi = 0.77351 \tag{9.16}$$

The standard deviation $\sigma(R) = 1.12$. Therefore, the value of R provides an estimate for the number of distinct elements N.

The hash function technique is very useful for those estimations in which non-repetitive elements have the same level of importance as repetitive elements. Some examples of such functions are those of finding distinct values [31, 43], mining inverse distributions [23], or determining the cardinality of set expressions [35]. The method in [43] uses a technique similar to that discussed in [31] in order to obtain a random sample of the distinct elements. This is then used for estimation. In [23], the problem of inverse distributions is discussed, in which it is desirable to determine the elements in the stream with a particular frequency level. Clearly such an inverse query is made difficult by the fact that a query for an element with very low frequency is equally likely to that of an element with very high frequency. The method in [23] solves this problem using a hash based approach similar to that discussed in [31]. Another related problem is that of finding the number of *distinct elements* in a join after eliminating duplicates. For this purpose, a join-distinct sketch (or JD-Sketch) was proposed in [36], which uses a 2-level adaptation of the hash function approach in [31].

4.7 Advantages and Limitations of Sketch Based Methods

One of the key advantages of sketch based methods is that they require space which is sublinear in the data size being considered. Another advantage of sketch based methods that it is possible to maintain sketches in the presence of deletions. This is often not possible with many synopsis methods such as random samples. For example, when the ith item with frequency $f(i)$ is deleted, the jth component of the sketch can be updated by subtracting $f(i) \cdot r_i^j$ from it. Another advantage of using sketch based methods is that they are extraordinarily space efficient, and require space which is logarithmic in the *number of distinct items* in the stream. Since the number of distinct items is significantly smaller than the size of the stream itself, this is an extremely low space requirement.

We note that sketches are based on the Lipshitz embeddings, which preserve a number of aggregate measures such as the L_p norm or the dot product. However, the entire distribution on the data (including the local temporal behavior) are not captured in the sketch representation, unless one is willing to work with a much larger space requirement.

Most sketch methods are based on analysis along a single dimensional stream of data points. Many problems in the data stream scenario are inherently multi-dimensional, and may in fact involve hundreds or thousands of independent and simultaneous data streams. In such cases, it is unclear whether sketch based methods can be easily extended. While some recent work in [16] provides a few limited methods for multi-dimensional queries, these are not easily extensible for more general problems. This problem is not however unique to sketch based methods. Many other summarization methods such as wavelets or histograms can be extended in a limited way to the multi-dimensional case, and do not work well beyond dimensionalities of 4 or 5.

While the concept of sketches is potentially powerful, one may question whether sketch based methods have been used for the right problems in the data stream domain. Starting with the work in [6], most work on sketches focuses on the *aggregate frequency* behavior of individual items rather than the temporal characteristics of the stream. Some examples of such problems are those of finding the frequent items, estimation of *frequency moments*, and *join size estimation*. The underlying assumption of these methods is an extremely large *domain size* of the data stream. The actual problems solved (aggregate frequency counts, join size estimation, moments) are relatively simple for modest domain sizes in many practical problems over very fast data streams. In these cases, temporal information in terms of sequential arrival of items is aggregated and therefore lost. Some sketch-based techniques such as those in [61] perform temporal analysis over specific time windows. However, this method has much larger space requirements. It seems to us that many of the existing sketch based methods can be easily extended to the temporal representation of the stream. It would be interesting to explore how these methods compare with other synopsis methods for temporal stream representation.

We note that the problem of aggregate frequency counts is made difficult only by the assumption of *very large domain sizes*, and not by the speed of the stream itself. It can be argued that in most practical applications, the data stream itself may be very fast, but the number of distinct items in the stream may be of manageable size. For example, a motivating application in [44] uses the domain of call frequencies of phone records, an application in which the number of distinct items is bounded above by the number of phone numbers of a particular phone company. With modern computers, it may even be possible to hold the frequency counts of a few million distinct phone numbers in a main memory array. In the event that main memory is not sufficient, many efficient

disk based index structures may be used to index and update frequency counts. We argue that many applications in the sketch based literature which attempts to find specific properties of the frequency counts (eg. second moments, join size estimation, heavy hitters) may in fact be implemented trivially by using simple main memory data structures, and the ability to do this will only increase over time with hardware improvements. There are however a number of applications in which hardware considerations make the applications of sketch based methods very useful. In our view, the most fruitful applications of sketch based methods lie in its recent application to the sensor network domain, in which in-network computation, storage and communication are greatly constrained by power and hardware considerations [22, 67, 68]. Many distributed applications such as those discussed in [9, 24, 70, 80] are particularly suited to this approach.

5. Histograms

Another key method for data summarization is that of histograms. In the method of histograms, we essentially divide the data along any attribute into a set of ranges, and maintain the count for each bucket. Thus, the space requirement is defined by the number of buckets in the histogram. A naive representation of a histogram would discretize the data into partitions of equal length (equi-width partitioning) and store the frequencies of these buckets. At this point, we point out a simple connection between the histogram representation and Haar wavelet coefficients. If we construct the wavelet representation of the *frequency distribution* of a data set along any dimension, then the (non-normalized) Haar coefficients of any order provide the difference in relative frequencies in equi-width histogram buckets. Haar coefficients of different orders correspond to buckets of different levels of granularity.

It is relatively easy to use the histogram for answering different kinds of queries such as range queries, since we only need to determine the set of buckets which lie within the user specified ranges [69, 81]. A number of strategies can be devised for improved query resolution from the histogram [69, 81, 84, 85].

The key source of inaccuracy in the use of histograms is that the distribution of the data points within a bucket is not retained, and is therefore assumed to be uniform. This causes inaccuracy because of extrapolation at the query boundaries which typically contain only a fractional part of a histogram. Thus, an important design consideration in the construction of histograms is the determination of how the buckets in the histogram should be designed. For example, if each range is divided into equi-width partitions, then the number of data points would be distributed very unequally across different buckets. If such buckets include the range boundary of a query, this may lead to inaccurate query estimations.

Therefore, a natural choice is to pick equi-depth buckets, in which each range contains an approximately equal number of points. In such cases, the maximum inaccuracy of a query is equal to twice the count in any bucket. However, in the case of a stream, the choice of ranges which would result in equi-depth partitions is not known a-priori. We note that the design of equi-depth buckets is exactly the problem of quantile estimation in data streams, since the equi-depth partitions define different quantiles in the data.

A different choice for histogram construction is that of minimizing the frequency variance of the different values within a bucket, so that the uniform distribution assumption is approximately held for queries. This minimizes the boundary error of extrapolation in a query. Thus, if a bucket B with count $C(B)$ contains the frequency of $l(B)$ elements, then average frequency of each element in the bucket is $C(B)/l(B)$. Let $f_1 \ldots f_{l(B)}$ be the frequencies of the l values within the bucket. Then, the total variance $v(B)$ of the frequencies from the average is given by:

$$v(B) = \sum_{i=1}^{l} (f_i - C(B)/l(B))^2 \tag{9.17}$$

Then, the total variance V across all buckets is given by the following:

$$V = \sum_{B} v(B) \tag{9.18}$$

Such histograms are referred to as *V-Optimal histograms*. A different way of looking at the V-optimal histogram is as a least squares fit to the frequency distribution in the data. Algorithms for V-Optimal histogram construction have been proposed in [60, 63]. We also note that the objective function to be optimized has the form of an L_p-difference function between two vectors whose cardinality is defined by the number of distinct values. In our earlier observations, we noted that sketches are particularly useful in tracking such aggregate functions. This is particularly useful in the multi-dimensional case, where the number of buckets can be very large as a result of the combination of a large number of dimensions. Therefore sketch-based methods can be used for the multi-dimensional case. We will discuss this in detail slightly later. We note that a number of other objective functions also exist for optimizing histogram construction [86]. For example, one can minimize the difference in the area between the original distribution, and the corresponding histogram fit. Since the space requirement is dictated by the number of buckets, it is also desirable to minimize it. Therefore, the dual problem of minimizing the number of buckets, for a given threshold on the error has been discussed in [63, 78].

One problem with the above definitions is that they use they use absolute errors in order to define the accuracy. It has been pointed out in [73] that the

use of absolute error may not always be a good representation of the error. Therefore, some methods for optimizing relative error have been proposed in [53]. While this method is quite efficient, it is not designed to be a data stream algorithm. Therefore, the design of relative error histogram construction for the stream case continues to be an open problem.

5.1 One Pass Construction of Equi-depth Histograms

In this section, we will develop algorithms for one-pass construction of equi-depth histograms. The simplest method for determination of the relevant quantiles in the data is that of sampling. In sampling, we simply compute the estimated quantile $q(S) \in [0, 1]$ of the true quantile $q \in [0, 1]$ on a random sample S of the data. Then, the Hoeffding inequality can be used to show that $q(S)$ lies in the range $(q - \epsilon, q + \epsilon)$ with probability at least $1 - \delta$, if the sample size S is chosen larger than $O(\log(\delta)/\epsilon^2)$. Note that this sample size is a constant, and is independent of the size of the underlying data stream.

Let v be the value of the element at quantile q. Then the probability of including an element in S with value less than v is a Bernoulli trial with probability q. Then the expected number of elements less than v is $q \cdot |S|$, and this number lies in the interval $(q^{\pm}\epsilon)$ with probability at least $2 \cdot e^{-2 \cdot |S| \cdot \epsilon^2}$ (Hoeffding inequality). By picking a value of $|S| = O(\log(\delta)/\epsilon^2)$, the corresponding results may be easily proved. A nice analysis of the effect of sample sizes on histogram construction may be found in [12]. In addition, methods for incremental histogram maintenance may be found in [42]. The $O(\log(\delta)/\epsilon^2)$ space-requirements have been tightened to $O(\log(\delta)/\epsilon)$ in a variety of ways. For example, the algorithms in [71, 72] discuss probabilistic algorithms for tightening this bound, whereas the method in [49] provides a deterministic algorithm for the same goal.

5.2 Constructing V-Optimal Histograms

An interesting offline algorithm for constructing V-Optimal histograms has been discussed in [63]. The central idea in this approach is to set up a dynamic programming recursion in which the partition for the last bucket is determined. Let us consider a histogram drawn on the N ordered distinct values $[1 \ldots N]$. Let $Opt(k, N)$ be the error of the V-optimal histogram for the first N values, and k buckets.Let $Var(p, q)$ be the variances of values indexed by p through q in $(1 \ldots N)$. Then, if the last bucket contains values $r \ldots N$, then the error of the V-optimal histogram would be equal to the sum of the error of the $(k-1)$-bucket V-optimal histogram for values up to $r - 1$, added to the error of the last bucket (which is simply the variance of the values indexed by r through N). Therefore, we have the following dynamic programming recursion:

$$Opt(k, N) = \min_r \{Opt(k - 1, r - 1) + Var(r, N)\} \tag{9.19}$$

We note that there are $O(N \cdot k)$ entries for the set $Opt(k, N)$, and each entry can be computed in $O(N)$ time using the above dynamic programming recursion. Therefore, the total time complexity is $O(N^2 \cdot k)$.

While this is a neat approach for offline computation, it does not really apply to the data stream case because of the quadratic time complexity. In [54], a method has been proposed to construct $(1 + \epsilon)$-optimal histograms in $O(N \cdot k^2 \cdot \log(N)/\epsilon)$ time and $O(k^2 \cdot \log(N)/\epsilon)$ space. We note that the number of buckets k is typically small, and therefore the above time complexity is quite modest in practice. The central idea behind this approach is that the dynamic programming recursion of Equation 9.19 is the sum of a monotonically increasing and a monotonically decreasing function in r. This can be leveraged to reduce the amount of search in the dynamic programming recursion, if one is willing to settle for a $(1 + \epsilon)$-approximation. Details may be found in [54]. Other algorithms for V-optimal histogram construction may be found in [47, 56, 57].

5.3 Wavelet Based Histograms for Query Answering

Wavelet Based Histograms are a useful tool for selectivity estimation, and were first proposed in [73]. In this approach, we construct the Haar wavelet decomposition on the cumulative distribution of the data. We note that for a dimension with N distinct values, this requires N wavelet coefficients. As is usually the case with wavelet decomposition, we retain the B Haar coefficients with the largest absolute (normalized) value. The cumulative distribution $\theta(b)$ at a given value b can be constructed as the sum of $O(\log(N))$ coefficients on the error-tree. Then for a range query $[a, b]$, we only need to compute $\theta(b) - \theta(a)$.

In the case of data streams, we would like to have the ability to maintain the wavelet based histogram dynamically. In this case, we perform the maintenance with frequency distributions rather than cumulative distributions. We note that when a new data stream element x arrives, the frequency distribution along a given dimension gets updated. This can lead to the following kinds of changes in the maintained histogram:

- Some of the wavelet coefficients may change and may need to be updated. An important observation here is that only the $O(\log(N))$ wavelet coefficients whose ranges include x may need to be updated. We note that many of these coefficients may be small and may not be included in the histogram in the first place. Therefore, only those coefficients which are already included in the histogram need to be updated. For a coefficient including a range of length $l = 2^q$ we update it by adding or subtracting $1/l$. We first update all the wavelet coefficients which are currently included in the histogram.

- Some of the wavelet coefficients which are currently not included in the histogram may become large, and may therefore need to be added to it. Let c_{min} be the minimum value of any coefficient currently included in the histogram. For a wavelet coefficient with range $l = 2^q$, which is not currently included in the histogram, we add it to be histogram with probability $1/(l * c_{min})$. The initial value of the coefficient is set to c_{min}.

- The addition of new coefficients to the histogram will increase the total number of coefficients beyond the space constraint B. Therefore, after each addition, we delete the minimum coefficient in the histogram.

The correctness of the above method follows from the probabilistic counting results discussed in [31]. It has been shown in [74] that this probabilistic method for maintenance is effective in practice.

5.4 Sketch Based Methods for Multi-dimensional Histograms

Sketch based methods can also be used to construct V-optimal histograms in the multi-dimensional case [90]. This is a particularly useful application of sketches since the number of possible buckets in the N^d space increases exponentially with d. Furthermore, the objective function to be optimized has the form of an L_2-distance function over the different buckets. This can be approximated with the use of the Johnson-Lindenstrauss result [64].

We note that each d-dimensional vector can be sketched over N^d-space using the same method as the AMS sketch. The only difference is that we are associating the 4-wise independent random variables with d-dimensional items. The Johnson-Lindenstrauss Lemma implies that the L_2-distances in the sketched representation (optimized over $O(b \cdot d \cdot \log(N)/\epsilon^2)$ possibilities) are within a factor $(1 + \epsilon)$ of the L_2-distances in the original representation for a b-bucket histogram.

Therefore, if we can pick the buckets so that L_2-distances are optimized in the sketched representation, this would continue to be true for the original representation within factor $(1 + \epsilon)$. It turns out that a simple greedy algorithm is sufficient to achieve this. In this algorithm, we pick the buckets greedily, so that the L_2 distances in the sketched representation are optimized in each step. It can be shown [90], that this simple approach provides a near optimal histogram with high probability.

6. Discussion and Challenges

In this paper, we provided an overview of the different methods for synopsis construction in data streams. We discussed random sampling, wavelets, sketches and histograms. In addition, many techniques such as clustering can

also be used for synopses construction. Some of these methods are discussed in more detail in a different chapter of this book. Many methods such as wavelets and histograms are closely related to one another. This chapter explores the basic methodology of each technique and the connections between different techniques. Many challenges for improving synopsis construction methods remain:

- While many synopses construction methods work effectively in individual scenarios, it is as yet unknown how well the different methods compare with one another. A thorough performance study needs to be conducted in understanding the relative behavior of different synopsis methods. One important point to be kept in mind is that the "trusty-old" sampling method provides the most effective results in many practical situations, where space is not constrained by specialized hardware considerations (such as a distributed sensor network). This is especially true for multi-dimensional data sets with inter-attribute correlations, in which methods such as histograms and wavelets become increasingly ineffective. Sampling is however ineffective in counting measures which rely on *infrequent* behavior of the underlying data set. Some examples are distinct element counting and join size estimation. Such a study may reveal the importance and robustness of different kinds of methods in a wide variety of scenarios.

- A possible area of research is in the direction of designing *workload aware* synopsis construction methods [75, 78, 79]. While many methods for synopsis construction optimize average or worst-case performance, the real aim is to provide optimal results for *typical* workloads. This requires methods for modeling the workload as well as methods for leveraging these workloads for accurate solutions.

- Most synopsis structures are designed in the context of quantitative or categorical data sets. It would be interesting to examine how synopsis methods can be extended to the case of different kinds of domains such as string, text or XML data. Some recent work in this direction has designed methods for XCluster synopsis or sketch synopsis for XML data [82, 83, 87].

- Most methods for synopsis construction focus on construction of optimal synopsis over the *entire data stream*. In many cases, data streams may evolve over time, as a result of which it may be desirable to construct optimal synopsis over specific time windows. Furthermore, this window may not be known in advance. This problem may be quite challenging to solve in a space-efficient manner. A number of methods for maintaining exponential histograms and time-decaying stream aggregates [15, 48]

try to account for evolution of the data stream. Some recent work on *biased reservoir sampling* [4] tries to extend such an approach to sampling methods.

We believe that there is considerable scope for extension of the current synopsis methods to domains such as sensor mining in which the hardware requirements force the use of space-optimal synopsis. However, the objective of constructing a given synopsis needs to be carefully calibrated in order to take the specific hardware requirements into account. While the broad theoretical foundations of this field are now in place, it remains to carefully examine how these methods may be leveraged for applications with different kinds of hardware, computational power, or space constraints.

References

[1] Aggarwal C., Han J., Wang J., Yu P. (2003) A Framework for Clustering Evolving Data Streams. *VLDB Conference*.

[2] Aggarwal C, Han J., Wang J., Yu P. (2004). On-Demand Classification of Data Streams. *ACM KDD Conference*.

[3] Aggarwal C. (2006) On Futuristic Query Processing in Data Streams. *EDBT Conference*.

[4] Aggarwal C. (2006) On Biased Reservoir Sampling in the Presence of Stream Evolution. *VLDB Conference*.

[5] Alon N., Gibbons P., Matias Y., Szegedy M. (1999) Tracking Joins and Self Joins in Limited Storage. *ACM PODS Conference*.

[6] Alon N., Matias Y., Szegedy M. (1996) The Space Complexity of Approximating the Frequency Moments. *ACM Symposium on Theory of Computing*, pp. 20–29/

[7] Arasu A., Manku G. S. Approximate quantiles and frequency counts over sliding windows. *ACM PODS Conference*, 2004.

[8] Babcock B., Datar M. Motwani R. (2002) Sampling from a Moving Window over Streaming Data. *ACM SIAM Symposium on Discrete Algorithms*.

[9] Babcock B., Olston C. (2003) Distributed Top-K Monitoring. *ACM SIGMOD Conference 2003*.

[10] Bulut A., Singh A. (2003) Hierarchical Stream summarization in Large Networks. *ICDE Conference*.

[11] Chakrabarti K., Garofalakis M., Rastogi R., Shim K. (2001) Approximate Query Processing with Wavelets. *VLDB Journal*, 10(2-3), pp. 199–223.

[12] Chaudhuri S., Motwani R., Narasayya V. (1998) Random Sampling for Histogram Construction: How much is enough? *ACM SIGMOD Conference*.

[13] Charikar M., Chen K., Farach-Colton M. (2002) Finding Frequent items in data streams. *ICALP*.

[14] Chernoff H. (1952) A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23:493–507.

[15] Cohen E., Strauss M. (2003). Maintaining Time Decaying Stream Aggregates. *ACM PODS Conference*.

[16] Cormode G., Garofalakis M., Sacharidis D. (2006) Fast Approximate Wavelet Tracking on Streams. *EDBT Conference*.

[17] Cormode G., Datar M., Indyk P., Muthukrishnan S. (2002) Comparing Data Streams using Hamming Norms. *VLDB Conference*.

[18] Cormode G., Muthukrishnan S. (2003) What's hot and what's not: Tracking most frequent items dynamically. *ACM PODS Conference*.

[19] Cormode G., Muthukrishnan S. (2004) What's new: Finding significant differences in network data streams. *IEEE Infocom*.

[20] Cormode G., Muthukrishnan S. (2004) An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. *LATIN* pp. 29-38.

[21] Cormode G., Muthukrishnan S. (2004) Diamond in the Rough; Finding Hierarchical Heavy Hitters in Data Streams. *ACM SIGMOD Conference*.

[22] Cormode G., Garofalakis M. (2005) Sketching Streams Through the Net: Distributed approximate Query Tracking. *VLDB Conference*.

[23] Cormode G., Muthukrishnan S., Rozenbaum I. (2005) Summarizing and Mining Inverse Distributions on Data Streams via Dynamic Inverse Sampling. *VLDB Conference*.

[24] Das A., Ganguly S., Garofalakis M. Rastogi R. (2004) Distributed Set-Expression Cardinality Estimation. *VLDB Conference*.

[25] Degligiannakis A., Roussopoulos N. (2003) Extended Wavelets for multiple measures. *ACM SIGMOD Conference*.

[26] Dobra A., Garofalakis M., Gehrke J., Rastogi R. (2002) Processing complex aggregate queries over data streams. *SIGMOD Conference*, 2002.

[27] Dobra A., Garofalakis M. N., Gehrke J., Rastogi R. (2004) Sketch-Based Multi-query Processing over Data Streams. *EDBT Conference*.

[28] Domingos P., Hulten G. (2000) Mining Time Changing Data Streams. *ACM KDD Conference*.

[29] Estan C., Varghese G. (2002) New Directions in Traffic Measurement and Accounting, *ACM SIGCOMM*, 32(4), *Computer Communication Review*.

[30] Fang M., Shivakumar N., Garcia-Molina H., Motwani R., Ullman J. (1998) Computing Iceberg Cubes Efficiently. *VLDB Conference*.

[31] Flajolet P., Martin G. N. (1985) Probabilistic Counting for Database Applications. *Journal of Computer and System Sciences*, 31(2) pp. 182–209.

[32] Feigenbaum J., Kannan S., Strauss M. Viswanathan M. (1999) An Approximate L_1-difference algorithm for massive data streams. *FOCS Conference*.

[33] Fong J., Strauss M. (2000) An Approximate L_p-difference algorithm for massive data streams. *STACS Conference*.

[34] Ganguly S., Garofalakis M., Rastogi R. (2004) Processing Data Stream Join Aggregates using Skimmed Sketches. *EDBT Conference*.

[35] Ganguly S., Garofalakis M, Rastogi R. (2003) Processing set expressions over continuous Update Streams. *ACM SIGMOD Conference*

[36] Ganguly S., Garofalakis M., Kumar N., Rastogi R. (2005) Join-Distinct Aggregate Estimation over Update Streams. *ACM PODS Conference*.

[37] Garofalakis M., Gehrke J., Rastogi R. (2002) Querying and mining data streams: you only get one look (a tutorial). *SIGMOD Conference*.

[38] Garofalakis M., Gibbons P. (2002) Wavelet synopses with error guarantees. *ACM SIGMOD Conference*.

[39] Garofalakis M, Kumar A. (2004) Deterministic Wavelet Thresholding with Maximum Error Metrics. *ACM PODS Conference*.

[40] Gehrke J., Korn F., Srivastava D. (2001) On Computing Correlated Aggregates Over Continual Data Streams. SIGMOD Conference.

[41] Gibbons P., Mattias Y. (1998) New Sampling-Based Summary Statistics for Improving Approximate Query Answers. *ACM SIGMOD Conference Proceedings*.

[42] Gibbons P., Matias Y., and Poosala V. (1997) Fast Incremental Maintenance of Approximate Histograms. *VLDB Conference*.

[43] Gibbons P. (2001) Distinct sampling for highly accurate answers to distinct value queries and event reports. *VLDB Conference*.

[44] Gilbert A., Kotidis Y., Muthukrishnan S., Strauss M. (2001) Surfing Wavelets on Streams: One Pass Summaries for Approximate Aggregate Queries. *VLDB Conference*.

[45] Gilbert A., Kotidis Y., Muthukrishnan S., Strauss M. (2003) One-pass wavelet decompositions of data streams. *IEEE TKDE*, 15(3), pp. 541–554. (Extended version of [44])

[46] Gilbert A., Kotidis Y., Muthukrishnan S., Strauss M. (2002) How to summarize the universe: Dynamic Maintenance of quantiles. *VLDB Conference*.

[47] Gilbert A., Guha S., Indyk P., Kotidis Y., Muthukrishnan S., Strauss M. (2002) Fast small-space algorithms for approximate histogram maintenance. *ACM STOC Conference*.

[48] Gionis A., Datar M., Indyk P., Motwani R. (2002) Maintaining Stream Statistics over Sliding Windows. *SODA Conference*.

[49] Greenwald M., Khanna S. (2001) Space Efficient Online Computation of Quantile Summaries. *ACM SIGMOD Conference*, 2001.

[50] Greenwald M., Khanna S. (2004) Power-Conserving Computation of Order-Statistics over Sensor Networks. *ACM PODS Conference*.

[51] Guha S. (2005). Space efficiency in Synopsis construction algorithms. *VLDB Conference*.

[52] Guha S., Kim C., Shim K. (2004) XWAVE: Approximate Extended Wavelets for Streaming Data. *VLDB Conference*, 2004.

[53] Guha S., Shim K., Woo J. (2004) REHIST: Relative Error Histogram Construction algorithms. *VLDB Conference*.

[54] Guha S., Koudas N., Shim K. (2001) Data-Streams and Histograms. *ACM STOC Conference*.

[55] Guha S., Harb B. (2005) Wavelet Synopses for Data Streams: Minimizing Non-Euclidean Error. *ACM KDD Conference*.

[56] Guha S., Koudas N. (2002) Approximating a Data Stream for Querying and Estimation: Algorithms and Performance Evaluation. *ICDE Conference*.

[57] Guha S., Indyk P., Muthukrishnan S., Strauss M. (2002) Histogramming data streams with fast per-item processing. *Proceedings of ICALP*.

[58] Hellerstein J., Haas P., Wang H. (1997) Online Aggregation. *ACM SIGMOD Conference*.

[59] Ioannidis Y., Poosala V. (1999) Histogram-Based Approximation of Set-Valued Query-Answers. *VLDB Conference*.

[60] Ioannidis Y., Poosala V. (1995) Balancing Histogram Optimality and Practicality for Query Set Size Estimation. *ACM SIGMOD Conference*.

[61] Indyk P., Koudas N., Muthukrishnan S. (2000) Identifying Representative Trends in Massive Time Series Data Sets Using Sketches. *VLDB Conference*.

[62] Indyk P. (2000) Stable Distributions, Pseudorandom Generators, Embeddings, and Data Stream Computation, *IEEE FOCS*.

[63] Jagadish H., Koudas N., Muthukrishnan S., Poosala V., Sevcik K., and Suel T. (1998) Optimal Histograms with Quality Guarantees. *VLDB Conference*.

[64] Johnson W., Lindenstrauss J. (1984) Extensions of Lipshitz mapping into Hilbert space. *Contemporary Mathematics*, Vol 26, pp. 189–206.

[65] Karras P., Mamoulis N. (2005) One-pass wavelet synopses for maximum error metrics. *VLDB Conference*.

[66] Keim D. A., Heczko M. (2001) Wavelets and their Applications in Databases. *ICDE Conference*.

[67] Kempe D., Dobra A., Gehrke J. (2004) Gossip Based Computation of Aggregate Information. *ACM PODS Conference*.

[68] Kollios G., Byers J., Considine J., Hadjieleftheriou M., Li F.(2005) Robust Aggregation in Sensor Networks. *IEEE Data Engineering Bulletin*.

[69] Kooi R. (1980) The optimization of queries in relational databases. *Ph. D Thesis*, Case Western Reserve University.

[70] Manjhi A., Shkapenyuk V., Dhamdhere K., Olston C. (2005) Finding (recently) frequent items in distributed data streams. *ICDE Conference*.

[71] Manku G., Rajagopalan S, Lindsay B. (1998) Approximate medians and other quantiles in one pass and with limited memory. *ACM SIGMOD Conference*.

[72] Manku G., Rajagopalan S, Lindsay B. (1999) Random Sampling for Space Efficient Computation of order statistics in large datasets. *ACM SIGMOD Conference*.

[73] Matias Y., Vitter J. S., Wang M. (1998) Wavelet-based histograms for selectivity estimation. *ACM SIGMOD Conference*.

[74] Matias Y., Vitter J. S., Wang M. (2000) Dynamic Maintenance of Wavelet-based histograms. *VLDB Conference*.

[75] Matias Y., Urieli D. (2005) Optimal workload-based wavelet synopsis. *ICDT Conference*.

[76] Manku G., Motwani R. (2002) Approximate Frequency Counts over Data Streams. *VLDB Conference*.

[77] Muthukrishnan S. (2004) Workload Optimal Wavelet Synopses. *DIMACS Technical Report*.

[78] Muthukrishnan S., Poosala V., Suel T. (1999) On Rectangular Partitioning in Two Dimensions: Algorithms, Complexity and Applications, *ICDT Conference*.

[79] Muthukrishnan S., Strauss M., Zheng X. (2005) Workload-Optimal Histograms on Streams. *Annual European Symposium*, Proceedings in *Lecture Notes in Computer Science*, 3669, pp. 734-745

[80] Olston C., Jiang J., Widom J. (2003) Adaptive Filters for Continuous Queries over Distributed Data Streams. *ACM SIGMOD Conference*.

[81] Piatetsky-Shapiro G., Connell C. (1984) Accurate Estimation of the number of tuples satisfying a condition. *ACM SIGMOD Conference*.

[82] Polyzotis N., Garofalakis M. (2002) Structure and Value Synopsis for XML Data Graphs. *VLDB Conference*.

[83] Polyzotis N., Garofalakis M. (2006) XCluster Synopses for Structured XML Content. *IEEE ICDE Conference*.

[84] Poosala V., Ganti V., Ioannidis Y. (1999) Approximate Query Answering using Histograms. *IEEE Data Eng. Bull.*

[85] Poosala V., Ioannidis Y., Haas P., Shekita E. (1996) Improved Histograms for Selectivity Estimation of Range Predicates. *ACM SIGMOD Conference*.

[86] Poosala V., Ioannidis Y. (1997) Selectivity Estimation without the Attribute Value Independence assumption. *VLDB Conference*.

[87] Rao P., Moon B. (2006) SketchTree: Approximate Tree Pattern Counts over Streaming Labeled Trees, *ICDE Conference*.

[88] Schweller R., Gupta A., Parsons E., Chen Y. (2004) Reversible Sketches for Efficient and Accurate Change Detection over Network Data Streams. *Internet Measurement Conference Proceedings*.

[89] Stolnitz E. J., Derose T., Salesin T. (1996) *Wavelets for computer graphics: theory and applications*, Morgan Kaufmann.

[90] Thaper N., Indyk P., Guha S., Koudas N. (2002) Dynamic Multidimensional Histograms. *ACM SIGMOD Conference*.

[91] Thomas D. (2006) Personal Communication.

[92] Vitter J. S. (1985) Random Sampling with a Reservoir. *ACM Transactions on Mathematical Software*, Vol. 11(1), pp 37–57.

[93] Vitter J. S., Wang M. (1999) Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. *ACM SIGMOD Conference*.