The VLDB Journal (2006) 15(1): 84–98
DOI 10.1007/s00778-004-0149-x

REGULAR PAPER

Umit Y. Ogras · Hakan Ferhatosmanoglu

Online summarization of dynamic time series data

Received: 18 March 2004 / Revised version: 9 October 2004 / Accepted: 9 October 2004 / Published online: 26 July 2005

Abstract Managing large-scale time series databases has attracted significant attention in the database community recently. Related fundamental problems such as dimensionality reduction, transformation, pattern mining, and similarity search have been studied extensively. Although the time series data are dynamic by nature, as in data streams, current solutions to these fundamental problems have been mostly for the static time series databases. In this paper, we first propose a framework to online summary generation for large-scale and dynamic time series data, such as data streams. Then, we propose online transform-based summarization techniques over data streams that can be updated in constant time and space. We present both the exact and approximate versions of the proposed techniques and provide error bounds for the approximate case. One of our main contributions in this paper is the extensive performance analysis. Our experiments carefully evaluate the quality of the online summaries for point, range, and *k–nn* queries using real-life dynamic data sets of substantial size.

Keywords Dimensionality reduction · Transformation-based summarization · Data streams · Time-series data

Edited by W. Aref

1 Introduction

Managing large-scale time series databases has attracted much attention in the database community. Current solutions mostly focus on the static version of the problem where the time series data are already stored in the database and is available for further processing. However, in many real-life applications involving time series data, such as data streams of stock tickers, network monitoring, and sensor networks, the data are modeled as dynamic time series, which are usually continuous and unbounded in nature. Since the data points arrive sequentially, storing each data point and performing an offline analysis is prohibitive and random access to the data is not allowed, unlike the traditional approaches. As a result, there is a need to develop novel, fast, online, and preferably one-pass analysis tools with small storage requirements for dynamic time series data.

Summaries of large databases have been utilized extensively in many applications. Specifically, transform-based methods have been widely used in databases to express the raw data in a coordinate system, where most of the energy is confined to a small subset of the transform coefficients. The summary, obtained by storing only the most significant transform coefficients, is used for many purposes including data mining, classification, clustering, selectivity estimation, query optimization, indexing, and efficient processing of queries including point, range, and similarity queries [1, 6, 28, 31, 33, 44]. Discrete Fourier transform (DFT) has been one of the most commonly used transformation techniques for a long time, also in the database literature [1, 13, 22, 41, 42, 48]. Similarly, discrete wavelet transform (DWT) [8], discrete cosine/sine transforms [29], and Karhunen–Loève transform (KLT) [32, 35, 43] have been applied to database systems, e.g., approximate query processing [13, 19, 24, 27, 38, 47] and selectivity estimation [34, 37].

The success of the transformation-based methods suggests that they can be applied to environments involving highly dynamic time series data, such as data streams, and by storing the summary in a small auxiliary space. Such an approach provides the ability to account for the previous data points with much smaller space requirements. Moreover, these summaries can be used in sensor networks and monitoring applications to analyze the rapidly evolving dynamic behavior of the environment [10, 36, 49]. However, developing the summary from scratch after the arrival of each element is very expensive. Hence, the application of the transformation-based methods to data streams depends

U. Y. Ogras (✉)
Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA
E-mail: uogras@andrew.cmu.edu

H. Ferhatosmanoglu
Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA
E-mail: hakan@cse.ohio-state.edu

critically on the development of efficient techniques to incrementally update the transform coefficients. Incremental generation of DWT-based summaries is discussed in [27]. The authors compute a sketch of the underlying data set to estimate the wavelet coefficients. The sketch of the data is updated incrementally as new data points are received. Then, the approximate wavelet-based summary is obtained as a batch process from this sketch. Similarly, maintenance histograms have been discussed in [26, 34, 38]. However, a general framework for dynamically maintaining the most significant transform coefficients of the time series data has not been explored. Moreover, there has been little work on extending the summarization techniques to sliding windows, and maintaining the top coefficients of the transform-based summaries for sliding window applications is open to discussion, as mentioned in [21].

In this paper, we develop a technique to dynamically maintain the transform-based synopsis of a data set based on the following observation. *Computing the transform-based synopsis of a data set is equivalent to finding the coefficients that minimize the squared error between the original sequence and the sequence reconstructed from the synopsis*. For static databases, directly transforming the data and keeping the desired coefficients is obviously cheaper than solving the least squares error (LSE) problem. On the other hand, transforming the data from scratch after the arrival of each element is prohibitive in applications where the data points arrive sequentially at a fast rate. Ideally, we want to find the synopsis as a function of the previous synopsis. The proposed technique constructs a fixed-size summary and maintains the summary efficiently using the observation mentioned above, as the data flows continuously.

Contributions of the paper We develop a general framework based on recursive least squares estimation (RLSE) to dynamically update the top M transform coefficients of streaming data. Then we extend the methodology for maintaining the synopsis of a sliding window of length N. The number of computations needed for maintaining the summary depends on the size of the synopsis, M, rather than the window size $N \gg M$. We illustrate this framework on DFT and show that the recursive computation produces the exact synopsis (the same result as transforming the data from scratch and keeping the top M coefficients) by using only $O(M)$ time and space. Extension to the sliding window case increases the space requirement to $O(N)$ for the maintenance of the exact synopsis. Hence, we devise an alternative technique that requires only $O(M)$ space, as well as $O(M)$ time, to generate an approximate synopsis with proven error bounds. Our final contribution is the extensive performance analysis. Our experiments with four real data sets demonstrate the following: (i) The proposed technique produces exact and approximate summaries with much greater efficiency than explicit computations, (ii) both the exact and approximate synopses keep most of the energy of the sliding window, and (iii) the summaries can be used to answer continuous and streaming queries with high performance.

The rest of the paper is organized as follows. Section 2 presents our framework. Section 3 illustrates the application of our framework to DFT. Section 4 presents a one-pass technique to generate approximate synopsis and derives the corresponding error bound. Section 5 includes an extensive evaluation and analysis of the proposed techniques using experiments on real data sets. Section 6 summarizes the related work including a detailed comparison to the wavelet-based summary generation method presented in [27]. Finally, Section 7 concludes the paper.

2 Development of the framework

In what follows, we first introduce our notation and then present the methodology for dynamically maintaining the synopsis of the time series data.

Stream model and notation We denote the time series data as a semi-infinite sequence $s(t)$, where t is an integer in the interval $[0, \infty]$. A sliding window covering N elements starting with the ith data point is denoted as $x_i = [s(i) : s(i+N-1)]$. Hence the most recent N elements can be expressed as $x_{t-N+1} = [s(t-N+1) : s(t)]$, where $s(t)$ is the most recent data point. Note that if $N = t$, the first window covers the whole data set, and x reduces to s. Hence, we represent the time series by x for both the general and sliding window case. The transformation of x, e.g., the DFT coefficients of x, is shown by $\mathcal{X}$. The transformation coefficients $\mathcal{X}$ has the same dimensionality as the original sequence x. Therefore, the original sequence can be perfectly recovered from $\mathcal{X}$. On the other hand, we denote the M-dimensional synopsis of x ($M \ll N$) by $\hat{\mathcal{X}}$. Finally, the window that can be reconstructed using this synopsis, which is an approximation to the original sequence, is denoted by $\hat{x}$. The models of incremental computation and sliding window are shown in Fig. 1. We assume that a large number of streams are continuously flowing into the system.

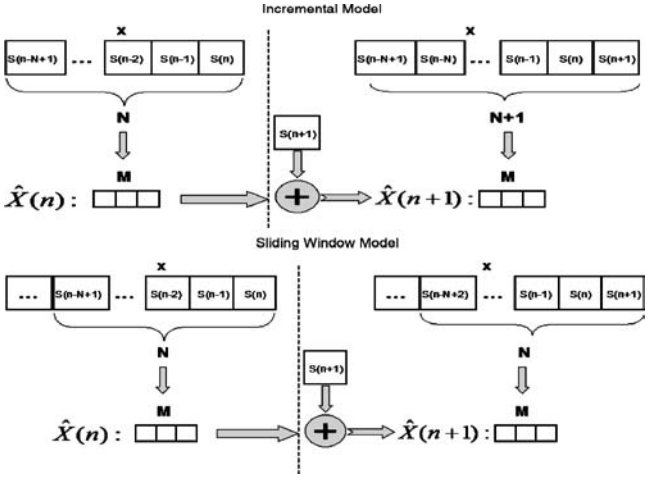

Fig. 1 Stream models

2.1 Recursive computation of the transform coefficients

An N-dimensional vector can be expressed by its transform coefficients and the basis vectors of the transformation $B = [b_0, b_1, \ldots, b_{N-1}]$ as $x = \sum_{i=0}^{N-1} \mathcal{X}(i) b_i$. Suppose that we want to approximate x using only $M < N$ basis vectors and denote the indices of these vectors by the set L. The approximation of x is obtained by a linear combination of the basis vectors as

$$\hat{x} = \sum_{i \in L} \hat{\mathcal{X}}(i) b_i \,.$$

The projection theorem [45] guarantees that the summary generated using this approach is optimum in the least squares sense, provided that $\hat{\mathcal{X}}(i) = \mathcal{X}(i)$ for $\forall i \in L$. Formally, it can be stated as follows. The sum of the squared errors between $x(n)$ and $\hat{x}(n)$

$$D^2(x, \hat{x}) = \sum_{t=0}^{N-1} [x(t) - \hat{x}(t)]^2 \tag{1}$$

is minimized if and only if

$$\hat{\mathcal{X}}(k) = \begin{pmatrix} \mathcal{X}(k) & k \in L \\ 0 & k \notin L \end{pmatrix} .$$

This result implies that we can determine the transform-based synopsis of a data set either by directly transforming the data (e.g., taking DFT, DCT, KLT, DWT etc.) and preserving the top coefficients or by finding the sequence $\hat{\mathcal{X}}(k)$ such that Eq. 1 is minimized. In the static case, the former way is preferred due to the higher computational efficiencies achieved using the fast transformation algorithms. However, when the data consist of a large number of rapid streams, we do not have sufficient processing time and space to store the data points and transform them. At this point, the recursive solution of LSE (RLSE) problem stands as an attractive alternative to computing the summary once and maintaining it continuously as the data flow. In what follows, we will represent the framework both formally and conceptually.

We start with a general model to solve the RLSE problem and extend it such that the new estimate is found recursively using the previous estimate and the new data point [39]. The following linear model can be used to estimate the synopsis of a sequence:

$$x(t) = H(t)\theta + v(t) \,.$$

In this model, $\theta \in \mathcal{C}^M$ is the synopsis that should be estimated. $x(t) \in \mathcal{R}^N$ stands for the current window and $H(t) \in \mathcal{C}^{N \times M}$ is the truncated inverse transformation matrix. $H(t)$ is obtained from the actual inverse transform matrix by preserving the columns corresponding to the M transform coefficients with the highest energy and deleting the remaining columns. Note that the linear system $x(t) = H(t)\theta$ has no solution since $M < N$. Hence the sequence $v(t)$ is added to satisfy the equality. This model is particularly useful when the number of data points is much larger than the number of parameters. This case is in fact what we are really interested in. Hence we consider the case $M \ll N$, where M is the dimension of the summary vector and N is the dimension of the data taken into account to generate the summary. Let the estimator of θ be $\hat{\mathcal{X}}$. Since we do not know θ itself, we cannot use $\|\theta - \hat{\mathcal{X}}\|$ as an objective function. Instead, the estimate of $x(t)$ is defined as $\hat{x}(t) = H(t)\hat{\mathcal{X}}$ and the objective function becomes

$$J[\hat{\mathcal{X}}] = [x(t) - \hat{x}(t)]^T W [x(t) - \hat{x}(t)] \,. \tag{2}$$

This objective function is simply the weighted sum of the squared errors between the real sample and its estimate. Note that this objective function is identical to the expression we want to minimize (Eq. 1), when W is the identity matrix. W is a weight matrix that reflects the relative importance of each sample. In applications where the recent data are believed to bear more information, a weight proportional to the age of the data can be assigned to each sample. The estimate that minimizes Eq. 2 is $\hat{\mathcal{X}} = (H^T W H)^{-1} H^T x(t)$. Remember that H is an $N \times M$ matrix, where N is probably a large number. Therefore, it is expensive to compute the estimate using this equation. On the other hand, it has a nice form that enables us to compute the estimate recursively as the new elements arrive. Let us assume that we have already observed N points and receive the $(N+1)$th point next. We express H as $H = \left[h_0^T, h_1^T, \ldots, h_{N-1}^T\right]^T$, where h_0 through h_{N-1} are the rows of H. The picture at time instant N is

$$\begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix} = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \end{bmatrix} \hat{\mathcal{X}}(N-1) + \begin{bmatrix} v(0) \\ v(1) \\ \vdots \\ v(N-1) \end{bmatrix} .$$

The index $(N-1)$ in expression $\hat{\mathcal{X}}(N-1)$ shows that the estimate is computed before the arrival of the Nth data point. The estimate after the addition of the next point is given in terms of the previous estimate and the latest data point by

$$\begin{aligned} \hat{\mathcal{X}}(N) = \hat{\mathcal{X}}(N-1) + P(N) h_N^T w(N) [x(N) \\ - h_N \hat{\mathcal{X}}(N-1)] \,, \end{aligned} \tag{3}$$

where

$$P(N)^{-1} = P(N-1)^{-1} + h_N^T w(N) h_N \tag{4}$$

$$P = (H^T W H)^{-1} \,, \tag{5}$$

as shown in [39]. After the first N data points arrive, the exact summary of the data is computed by a batch process. Then, the initial P matrix is computed using Eq. 5, and the summary is updated after the addition of each data point as follows: Eq. 4 updates P based on its previous value. Then Eq. 3 updates the summary using the previous estimate, the new P matrix, the new data point, and the inverse transformation matrix. Hence we use the recursive Eqs. 3 and 4 to compute the desired transform coefficients of the stream incrementally. Illustration of the maintenance

technique is left to Sect. 3, where we specifically consider DFT-based synopsis generation.

Maintaining the synopsis of a sliding window is more challenging because it requires that the oldest element in the window be dropped after a new element is received. In this case, we reflect the change in the window to the transform coefficients and update the synopsis. This is done as follows. We circularly shift the current window. The oldest element that has to leave the window becomes the first element after this operation. Then we replace this element with the most recent data point and reflect the effect of shifting and replacement operations to the transform. Storing the complete sliding window is the only way to account for the oldest element perfectly. Our framework produces the exact synopsis of each window, if we are allowed to store the whole sliding window. For other applications, where this is prohibitive, we approximate the element leaving the window accurately using the current synopsis and produce an approximate summary with proven error bounds. The following section illustrates the framework by dynamically maintaining DFT-based summaries.

3 Incremental DFT computation

In this section, we illustrate the framework developed in the preceding section by solving the following problem:

Problem definition Given a large number of dynamic time series data, dynamically maintain the DFT-based synopsis of the latest N points of each stream.

The DFT analysis and synthesis equations are $\mathcal{X}(k) = \sum_{t=0}^{N-1} x(t) W_N^{kt}$ and $x(t) = \frac{1}{N} \sum_{k=0}^{N-1} \mathcal{X}(k) W_N^{-kt}$, respectively, where $W_N = e^{\frac{-j2\pi}{N}}$. The synthesis equation expresses a sequence of length N as a sum of complex exponentials at frequencies $\frac{2\pi k}{N}$, where k is an integer, ranging from zero to $N-1$. In database applications, the lower-dimensional summary of data is obtained by truncating $\mathcal{X}(k)$ such that only the elements with the highest energy are retained. Although we are not interested in recovering the original data set from its summary, the similarity of the recovered data with the original set is used as a measure of the summary quality.

We represent the DFT operation as a matrix multiplication $\mathcal{X} = Tx$, where

$$T_{ij}] = \left[W_N^{-(i-1)(j-1)}\right], \quad i, j = 1, \ldots, N\,.$$

Since we are interested in the summary of the data, we want to keep only the top M coefficients of the DFT while setting the remaining ones to zero. Without loss of generality, during the derivation we assume that the leading M coefficients have the highest energy, as is the case for many real data sets [1]. The selection of these coefficients, which is made based on our knowledge about the data of interest, can be changed adaptively without affecting the derivation. We express the relation between the data set and its DFT-based synopsis as

$$x(t) = H\hat{\mathcal{X}}(k) + v(t)\,,$$

where $\hat{\mathcal{X}}$ is the $M \times 1$ synopsis and H is the $N \times M$ truncated DFT matrix obtained by retaining the basis vectors corresponding to the desired coefficients, as shown below.

$$H = \frac{1}{N} \begin{pmatrix} 1 & 1 & 1 & \cdots & \cdots & 1 & 1 \\ 1 & W_N^{-1.1} & \cdots & W_N^{-1.\frac{M-1}{2}} & W_N^{-1.(N-\frac{M-1}{2})} & \cdots & W_N^{-1.(N-1)} \\ 1 & W_N^{-2.1} & \cdots & W_N^{-2.\frac{M-1}{2}} & W_N^{-2.(N-\frac{M-1}{2})} & \cdots & W_N^{-2.(N-1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & W_N^{-(N-2).1} & \cdots & W_N^{-(N-2).\frac{M-1}{2}} & W_N^{-(N-2).(N-\frac{M-1}{2})} & \cdots & W_N^{-(N-2).(N-1)} \\ 1 & W_N^{-(N-1).1} & \cdots & W_N^{-(N-1).\frac{M-1}{2}} & W_N^{-(N-1).(N-\frac{M-1}{2})} & \cdots & W_N^{-(N-1).(N-1)} \end{pmatrix}$$

Note that the columns of T corresponding to the zero coefficients are deleted. As shown in Sect. 2, the synopsis $\hat{\mathcal{X}}$ that minimizes the objective function (Eq. 2) is the truncated DFT of the sequence $x(t)$. Since there is a model for solving this minimization problem recursively, we can compute and update the DFT coefficients dynamically as the new elements arrive. As an example, assume that 5 data points have already arrived and we need to dynamically update the DFT of the incoming stream. First, we compute the DFT of the sequence using the DFT analysis equation. Having obtained the initial summary, we compute the P matrix using Eq. 5. Then, the DFT coefficients are updated using Eqs. 3 and 4 after the arrival of each new data point. The initial number of data points, which is 5 in this simple example, is selected such that the batch process requires affordable time and storage. In our experiments, this number is selected as the size of the synopsis itself.

In what follows, we focus on maintaining the synopsis of a sliding window of size N. This is a more general problem than the incremental computation of the synopsis since the element leaving the window has to be accounted for in addition to the new data point. We can arbitrarily increase N to take all data points into account, as our update rule depends on the synopsis size rather than N.

Suppose a window consisting of N data points has already arrived and its most significant coefficients have been computed. We start with the summary, $\hat{\mathcal{X}}$, obtained for the first N points and update it as the new data points arrive. The picture at time instant N is

$$\begin{bmatrix} x(0) \\ \vdots \\ x(N-1) \end{bmatrix} = \begin{bmatrix} h_0 \\ \vdots \\ h_{N-1} \end{bmatrix} \hat{\mathcal{X}}(N-1) + \begin{bmatrix} v(0) \\ \vdots \\ v(N-1) \end{bmatrix}. \quad (6)$$

Unlike the scenario in Sect. 2.1, in sliding windows the oldest element is dropped when a new element is received. To take this fact into account, we make some manipulations. Let us take $x(0)$ and append it to the bottom. To preserve the equality, we also shift the rows of H and obtain

$$\begin{bmatrix} x(1) \\ \vdots \\ x(N-1) \\ x(0) \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ h_{N-1} \\ h_0 \end{bmatrix} \hat{\mathcal{X}}(N-1) + \begin{bmatrix} v(1) \\ \vdots \\ v(N-1) \\ v(0) \end{bmatrix}.$$

Since we have only applied row operations to Eq. 6, $\hat{\mathcal{X}}(N-1)$ remains unchanged. Note that this estimate could have been computed by appending $x(0)$, h_0, and $v(0)$ to the $(N-1)$-dimensional system given by

$$\begin{bmatrix} x(1) \\ \vdots \\ x(N-1) \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ h_{N-1} \end{bmatrix} \hat{\mathcal{X}}_{\text{aux}} + \begin{bmatrix} v(1) \\ \vdots \\ v(N-1) \end{bmatrix}.$$

Using Eq. 3 and the hypothetical estimate $\hat{\mathcal{X}}_{\text{aux}}$ (an auxiliary estimate to be eliminated), we can express $\hat{\mathcal{X}}_{N-1}$ as

$$\hat{\mathcal{X}}(N-1) = \hat{\mathcal{X}}_{\text{aux}} + P(N)h_0^T[x(0) - h_0\hat{\mathcal{X}}_{\text{aux}}].$$

Now, assume that we append $x(N)$ instead of $x(0)$ to the sequence (i.e., the window slid one point and covers $[x(1), x(2), \ldots, x(N)]$). In that case, the new estimate would be

$$\hat{\mathcal{X}}(N) = \hat{\mathcal{X}}_{\text{aux}} + P(N)h_0^T[x(N) - h_0\hat{\mathcal{X}}_{\text{aux}}].$$

If we let $x(N) - x(0) = \Delta_N$ and substitute $x(N)$ with $\Delta_N + x(0)$, we can obtain $\hat{\mathcal{X}}(N)$ in terms of $\hat{\mathcal{X}}(N-1)$ as $\hat{\mathcal{X}}(N) = \hat{\mathcal{X}}(N-1) + P(N)h_0^T\Delta_N$. In general, the synopsis of the nth window is given by

$$\hat{\mathcal{X}}_n = \hat{\mathcal{X}}_{n-1} + P(N)h_0^T\Delta_n\,.$$

This equation gives a simple update rule to estimate $\hat{\mathcal{X}}_n$ using the previous estimate $\hat{\mathcal{X}}_{n-1}$. It only requires the difference between the last point and the point leaving the window, $h_0 = [1, 1, \ldots, 1]_{M\times 1}$ and $P(N)$. However, the updated estimate is not the DFT of the new window because H is distorted by changing the rows. Consequently, it is not equal to the matrix H given in Table 1. If we can find a matrix A such that

$$\left[h_1^T, \ldots, h_{N-1}^T, h_0^T\right]^T A = H = \left[h_0^T, h_1^T, \ldots, h_{N-1}^T\right]^T,$$

then we can find DFT coefficients from $\hat{\mathcal{X}}_n$. That is, we want

$$\begin{bmatrix} x(1) \\ \vdots \\ x(N-1) \\ x(N) \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ h_{N-1} \\ h_0 \end{bmatrix} AA^{-1}\hat{\mathcal{X}}_n + \begin{bmatrix} v(1) \\ \vdots \\ v(N-1) \\ v(N) \end{bmatrix},$$

where $A^{-1}\hat{\mathcal{X}}_n$ will give the desired DFT coefficients. Since we do not need to find A itself to compute the DFT, we directly compute A^{-1}. Matrices A and P take very convenient forms for DFT. P is found as $P = N \times I_{k\times k}$ and A turns out to be a diagonal matrix

$$A^{-1} = \text{diag}\big(1, W_N^{-1}, \ldots, W_N^{-\frac{M-1}{2}}, \ldots, W_N^{-(N-\frac{M-1}{2})}, \ldots, W_N^{-(N-2)}, W_N^{-(N-1)}\big). \quad (7)$$

The details of the computations are provided in the appendix.

To sum up, we obtain the DFT of the nth window in terms of the previous window as

$$\hat{\mathcal{X}}_n = A^{-1}\hat{\mathcal{X}}_{n-1} + \frac{A^{-1}}{N}h_0^T(x(N) - x(0)). \quad (8)$$

The first operation in Eq. 8 requires only M multiplications and additions since A^{-1} is a diagonal matrix given in Eq. 7. The second operation also requires only M multiplications since $h_0 = [1, 1, \ldots, 1]_{M\times 1}$. So a total of $2M$ multiplications and M additions are needed. This is much more efficient than computing the DFT of each window explicitly since the number of coefficients stored is much less than the size of the window ($M \ll N$).

The method presented in this section produces the exact synopsis with computational complexity depending only on the synopsis size M. However, it requires that the sliding window be stored to access the window's oldest element. This may be prohibitive in applications where the size of the data to be processed is very large. Hence, it may be desirable to trade accuracy with space for these applications. In the following section, we derive an algorithm to produce one-pass, approximate synopses with corresponding error bounds.

4 Approximate synopsis generation

There is a tradeoff between the accuracy of the synopsis and the space requirement of the method. Computation of the exact synopsis requires both the newest data point, $x(N)$, and the oldest element, $x(0)$, that leaves the window. Instead of storing the complete window for just one value and accessing it during the update process, we can use the current synopsis to approximate $x(0)$. The resulting algorithm has the one-pass property besides being highly efficient in time and space.

The inverse DFT of the synopsis is given by

$$\hat{x}(t) = \frac{1}{N}\sum_{k=0}^{N-1}\hat{\mathcal{X}}(k)W_N^{-kt}\,.$$

Since we are interested in the first value in the window, we do not need to compute the entire inverse DFT. Instead, we substitute $t = 0$ and find only $\hat{x}(0)$ as

$$\hat{x}(0) = \frac{1}{N}\sum_{k=0}^{N-1}\hat{\mathcal{X}}(k) = \frac{1}{N}\sum_{k=0}^{M-1}\mathcal{X}(k)\,. \quad (9)$$

In the above equation, we use the fact that the first M coefficients of $\hat{\mathcal{X}}$ are equal to those of X and the remaining are zero. In this way, we can update the synopsis using only the previous synopsis and the newest data point. Hence, the space requirement of the algorithm drops dramatically at the expense of the error introduced by the estimation. It is necessary to keep the error within acceptable bounds to maintain the quality of the results. Therefore, we analyze the error and find an upper bound in the next section.

4.1 An error bound for the approximation

The update rule to generate the synopsis of the nth window in terms of the previous synopsis and the latest data point is given by Eq. 8. If we use the estimate of $x(0)$ instead of the actual value, the approximate update is found as

$$\bar{\mathcal{X}}_n = A^{-1}\bar{\mathcal{X}}_{n-1} + \frac{A^{-1}}{N}h_0^T\left(x(N) - \hat{x}(0)\right). \tag{10}$$

Hence the error in the synopsis evaluated at the nth time instant is found by subtracting Eq. 10 from Eq. 8:

$$\tilde{\mathcal{X}}_n = A^{-1}\tilde{\mathcal{X}}_{n-1} + \frac{A^{-1}}{N}h_0^T\left(\hat{x}(0) - x(0)\right). \tag{11}$$

This equation gives the error in the nth synopsis in terms of the error in the previous synopsis. Since we know that $\tilde{\mathcal{X}}_0 = 0$, we can rewrite Eq. 11 as

$$\begin{aligned}\tilde{\mathcal{X}}_n &= \frac{(A^{-1})^n h_0^T}{N}\Delta x_0 + \frac{(A^{-1})^{n-1}h_0^T}{N}\Delta x_1 \\ &\quad + \cdots + \frac{A^{-1}h_0^T}{N}\Delta x_{n-1} \\ \tilde{\mathcal{X}}_n &= \frac{1}{N}[(A^{-1})^n h_0^T | \cdots | A^{-1}h_0^T]\begin{bmatrix}\Delta x_0\\ \Delta x_1\\ \vdots\\ \Delta x_{n-1}\end{bmatrix}, \end{aligned} \tag{12}$$

where $\Delta x_n = \hat{x}_n(0) - x_n(0)$. Since A^{-1} is diagonal, $(A^{-1})^n$ can be found as

$$(A^{-1})^n = \mathrm{diag}\left(1, W_N^{-n}, W_N^{-2n}, \ldots, W_N^{-(N-1)n}\right).$$

As a result, $\|(A^{-1})^n h_0^T | \cdots | A^{-1}h_0^T\|_1 = M$. If we take the norm of both sides of Eq. 12, we obtain

$$\|\tilde{\mathcal{X}}_n\|_1 \le \frac{M}{N}\sum_{l=0}^{n-1}\|\Delta x_l\|. \tag{13}$$

Note that $\Delta x_l = -\frac{1}{N}\sum_{k=M}^{N-1}\mathcal{X}_l(k)$. Since we take the top M coefficients, we can write

$$\|\Delta x_l\| \le \frac{N-M}{N}\|\mathcal{X}_l(M-1)\|.$$

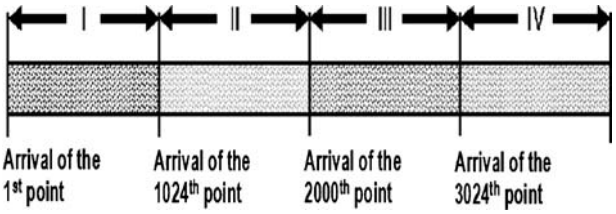

Fig. 2 Maintaining the approximate synopsis

Hence we obtain the following upper bound for the error

$$\|\tilde{\mathcal{X}}_n\| \le \frac{M(N-M)}{N^2}\sum_{l=0}^{n-1}\|\mathcal{X}_l(M-1)\|. \tag{14}$$

The error bound given in Eq. 14 is a function of the synopsis and window sizes. As M approaches the window size N, i.e., we take more coefficients, the bound goes to zero as expected. Similarly, the error is accumulating as more elements are received and the synopsis is updated by approximating $x(0)$. In practical applications, the theoretical bound can be tracked, and when it exceeds a threshold value, the synopsis can be reconstructed again. We propose to reset the synopsis (i.e., replacing the approximate synopsis with the exact one) periodically to keep the error always within a very small range, as illustrated in Fig. 2. Suppose that we select the window size as 1,024 and our reset rate as 2,000. The algorithm goes as follows:

Period I The exact synopsis is constructed incrementally as the data points arrive using Eqs. 3 and 4. The queries are answered using the exact synopsis.

Period II The exact synopsis, obtained after the arrival of the 1,024th data point, is used as the initial summary. Then, the approximate synopsis is computed after the arrival of each data point. The queries are answered using the approximate synopsis.

Period III During this period, two processes run concurrently. The first one is the computation of the approximate synopsis as in period II. We continue to answer the queries using this approximate synopsis. The second process is the construction of an exact synopsis assuming that the 2,001th data point is the first element in the window (the same process as performed in period I). This concurrency is affordable since both tasks are performed very efficiently.

Period IV At the end of period III, there are two synopses for the same sliding window. The first one is the approximate synopsis used during periods II and III, while the other one is the exact synopsis whose construction started at the beginning of period III. At the beginning of period IV, we drop the approximate synopsis and proceed exactly as period II. After that, periods III and IV are repeated in an alternating sequence. Hence, the patten is: I, II, III, IV, III, IV, ...

Such an approach enables us to maintain very accurate summaries of a large number of time series data originating from different sources using only $O(M)$ space for each series. In the experiments, we show that a period in the order of the window size can keep the error within 10% of the exact synopsis.

5 Experiments

In the preceding sections, we developed a recursive method to compute and maintain exact and approximate summaries of time series data. In this section, we perform a complete set of experiments to demonstrate the efficiency of our technique and the accuracy of the results. The experiments are performed on a standard machine equipped with 512 MB memory and a Pentium 4 processor operating at 2.26 GHz. The proposed techniques are implemented in Matlab 6.1. The first set of experiments show that our techniques achieve high caliber results with superior performance compared to explicit DFT computation. The second set of experiments illustrates usefulness of the summaries in answering continuous and streaming queries.

5.1 Performance of synopsis generation

Time and space requirements of the technique and the quality of its results are the main criteria in assessing the performance. For this reason, we run experiments to explore these aspects in a complete manner. In these experiments, we assumed that there was a large number of sources supplying time series data. We analyzed the effects of the sliding window size, the synopsis size, and the approximation error on the efficiency by employing four real data sets.

5.1.1 Computation time

The first experiment illustrates the fact that the proposed exact synopsis generation technique produces the same result as taking the DFT of each window explicitly and retaining the desired coefficients, with much higher efficiency. We used a time series data set representing stock market movements of 32,500 companies over 1,080 periods, i.e., there are 32,500 different streams flowing to our system and we continue the experiment until 1,080 data elements are received. We selected the synopsis size as 11 and varied the window size from 16 to 750.

Figure 3a shows the total time it takes to update the synopses of all 32,500 different series. In this figure, the x-axis

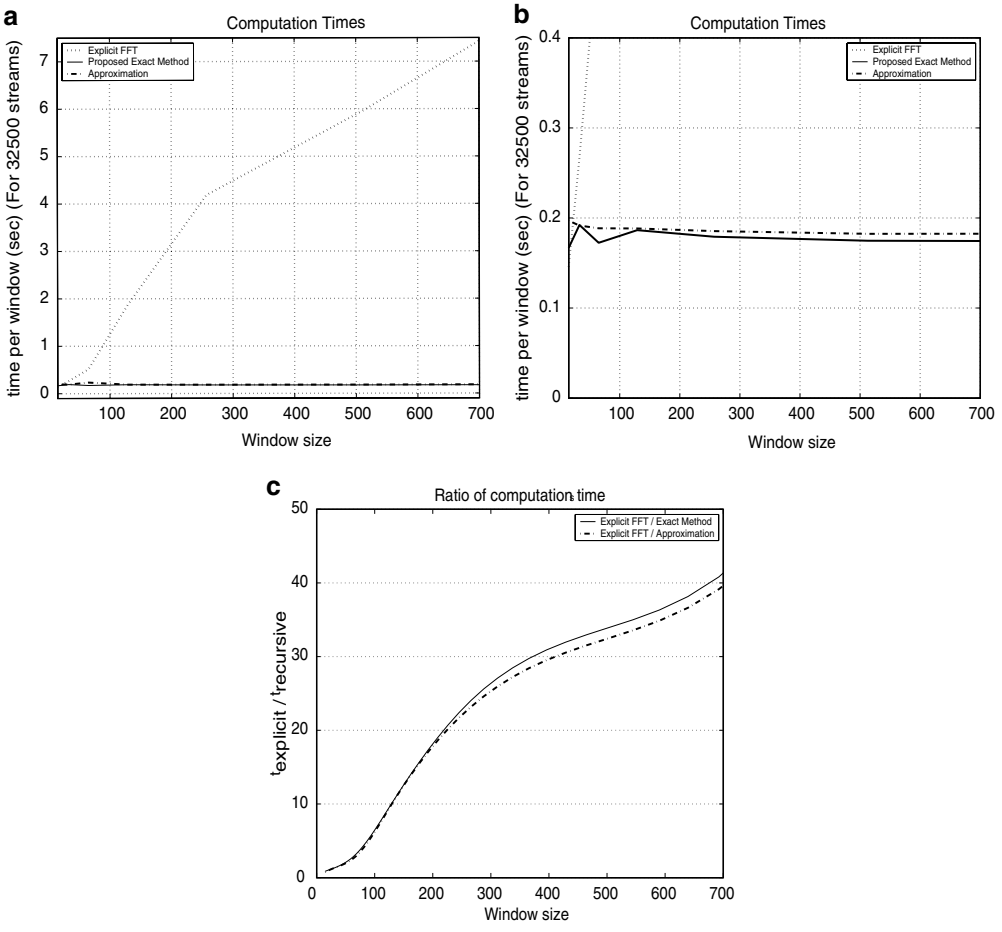

Fig. 3 **a** Time it takes to update the synopsis for each window of 32,500 time series data sets. **b** Performance of the proposed techniques are independent of window size. Approximation causes a slight increase in computation time. **c** The ratio of the computation times of the explicit DFT and our methods multiples for increasing window size since explicit DFT requires $N \log N$ time

is the window size while the y-axis shows the total computation time of the 32,500 DFTs for the corresponding window sizes (computation time per window). Our method clearly outperforms explicit computation. To analyze the performance of the proposed techniques more clearly, we provide the zoomed version of this plot in Fig. 3b. It takes about 190 ms for the proposed technique to compute the synopses of all series. This means that the proposed technique can handle about 171,050 streams per second with a nonoptimized code on a standard machine. While this rate is sufficient for many applications, the capacity can be further increased using other newly proposed techniques, such as smart sampling and shedding [4, 15, 17, 30]. Figure 3b also shows that the time required to update the DFT coefficients using our approach remains constant as N increases. This is expected since the update process is a function of the synopsis size rather than the window size. The approximation, which needs M more additions and one division, takes slightly more time, as expected. Finally, we present the ratio of the computation times of the explicit DFT and our method in Fig. 3c. Our technique performs about 34 times faster for window size $N = 512$ and the ratio increases with the window size. Further experiments showed that the computation time, indeed, increases linearly with increasing synopsis size as predicted by Eq. 8.

5.1.2 The quality of the synopsis

We evaluated the quality of our results following a systematic approach. First, we compared the summaries with the complete description of the data and justified that they preserve most of the energy of the original data set. After verifying that the exact synopsis was indeed a good approximation, we compared it with the approximate synopsis. These experiments demonstrate that our one-pass technique generates accurate summaries of the original data. Furthermore, they provide an explanation for the outstanding query performance of the resulting summaries.

We start the experiments with the same time series data of size $1{,}080 \times 32{,}500$. In this part, we fix the window size to 128. First, we compute the total energy of the ith window, given by $x = [s(i) : s(i+127)]$, of each stream by assuming that we already had them. Then, the sum of the squared errors (SSE) between each window and its synopsis are found both for the exact and approximate summaries. Finally, the results are averaged over all different time series. The set of plots given in Fig. 4 shows that most of the energy of the original data set is preserved by the exact synopsis. The ratio decreases, i.e., more energy is confined in the synopsis, as we enlarge the synopsis size. We also compute and plot the corresponding results for the approximate synopsis. While the approximate synopsis results in a higher ratio, it still captures more than 90% of the energy for $M = 5$. As the procedure is repeated at different time instances, we are able to observe the variation of the performance with time. In particular, we show the results for the windows 20, 60, 100, 140, 180, and 220 in Fig. 4. The performance of

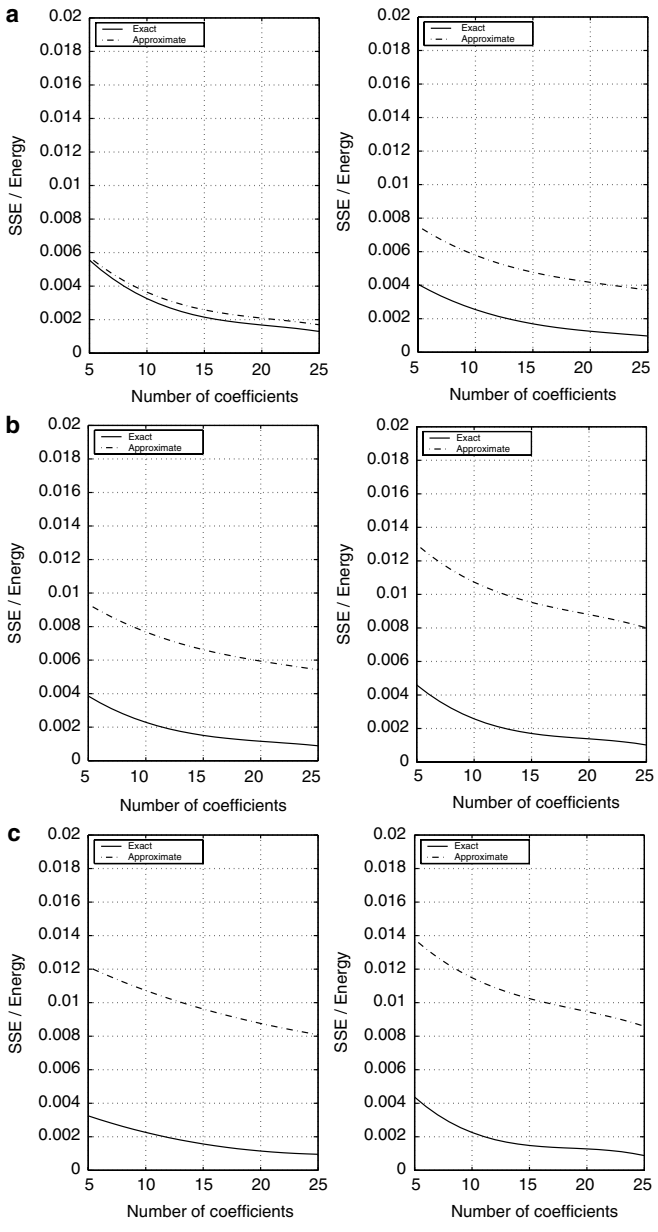

Fig. 4 Ratio of SSE and energy of original data for varying synopsis sizes. The SSE decreases as more coefficients are retained in the synopsis. The increase of the SSE for the approximate synopsis as window number increases is illustrated by plotting the results for a series of windows. **a** 20th and 60th windows. **b** 100th and 140th windows. **c** 180th and 220th windows

the exact synopsis is almost the same for all of the windows. On the other hand, the performance of the approximate synopsis degrades as time passes. This is expected, because we have proven that the error in this result accumulates due to the approximation of the oldest data point in each window.

Next, we analyze the error caused by the approximation in our one-pass synopsis generation technique. For this case, we compute the difference between the exact and approximate synopses of size 11 and find the sum of the squared

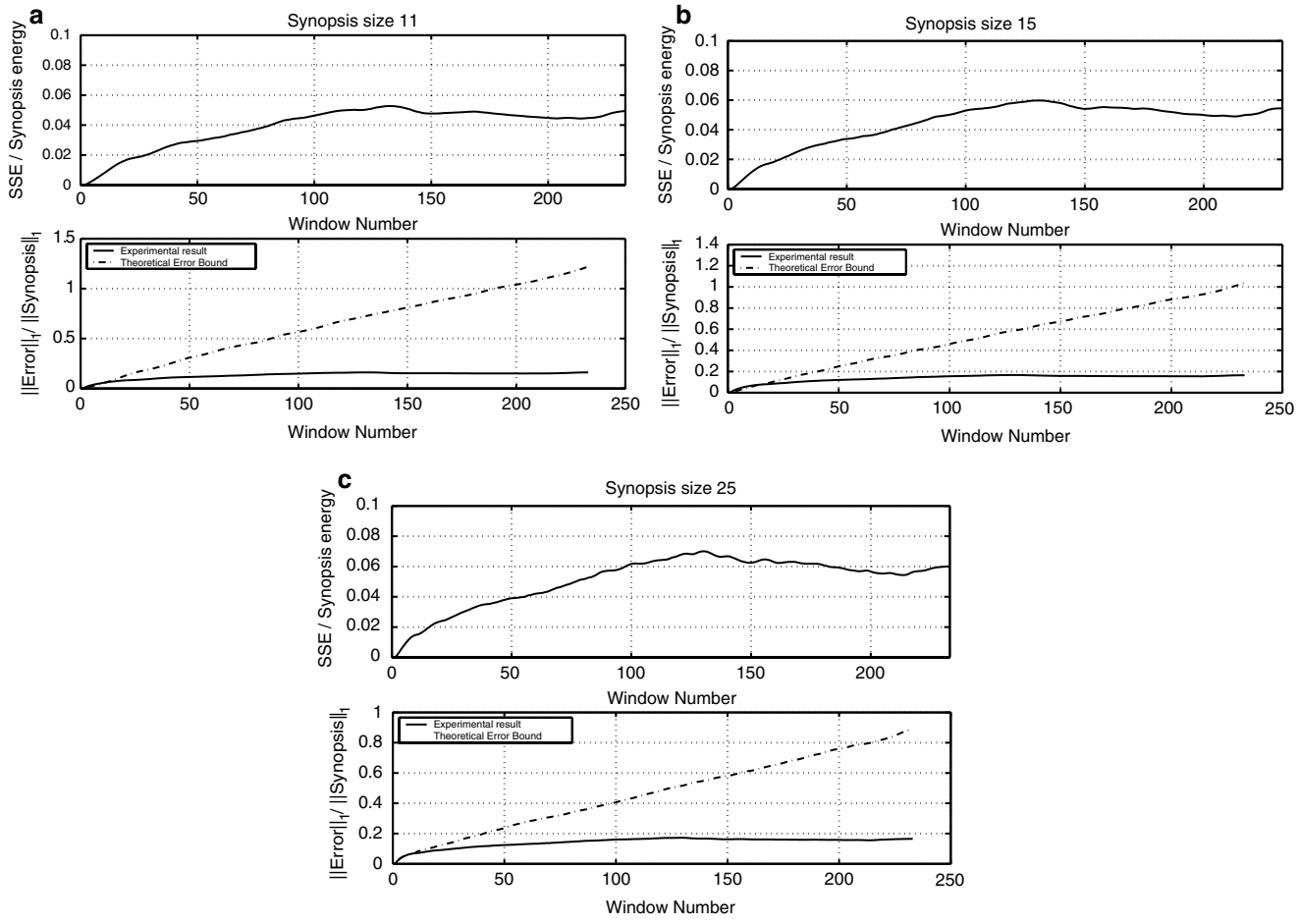

Fig. 5 The ratio of the SSE between the exact and approximate summaries, and the energy of the exact synopsis are given in the upper plots. The *lower plots* show the same ratio for the L_1-norms and the theoretical error bound. **a** Synopsis size 11. **b** Synopsis size 15. **c** Synopsis size 25

errors (SSE). Then these results are divided by the energy of the exact synopsis. We plot the results for windows 1–232 as shown in Fig. 5. It is observed that the ratio increases as the window slides over the data. However, the rate of increase reduces to zero. The same procedure is repeated for different synopsis sizes, as shown in Fig. 5. We also compute the ratio of the L_1-norm of the error and the exact synopsis and compare it with the theoretical error bound. We observe that it is indeed below the theoretical bound and its behavior is similar to the ratio of energies. We repeat these experiments with isolated letter speech recognition data of size $617 \times 2{,}000$ and obtain similar results.

To analyze the effect of the error accumulation better, we used other real data sets with much larger number of dimensions. The results obtained with 180,000 dimensional electrocardiogram (ECG) data recorded from a human (male) in a supine position are shown in Fig. 6. Figure 6a–c gives the ratio of SSE and the energy of the original data set for varying synopsis sizes. The ratio diminishes as more coefficients are retained in the summary as for the previous data set. The variation of SSE between exact and approximate summaries is illustrated in Fig. 6d–f. Unlike the previous experiment, we reset the approximate synopsis with period 512 as explained in Sect. 4. As can be seen, the error accumulates as for the previous data set. However, when we replace the approximate with the exact synopsis, the error, and hence the ratio, drops to zero. In this way, we manage to keep the approximate synopsis within 90% of the exact synopsis. Similarly, we repeat the same experiments for a data set of length 26,612 (Dow Jones Industrial Average between 1900 and 1993). The results obtained with synopsis size 9 for this data set are provided in Fig. 7. In this case, the window size is selected as 2,400. In Fig. 7a–c, it is observed that the exact and approximate summaries keep more than 99% of the total energy for a synopsis size larger than 9. In Fig. 7d,e, we see that the approximate synopsis captures more than 95% of the exact synopsis for most of the windows, while for some of the windows this ratio drops to 80%.

If the error is crucial, the reset rate can be make as low as the window size. In this case, the number of data streams the technique can handle will be halved, since the update of the approximate synopsis and construction of the exact one that will replace it will be performed concurrently. Hence, the data rate will be 85,525 streams/s with the current setup.

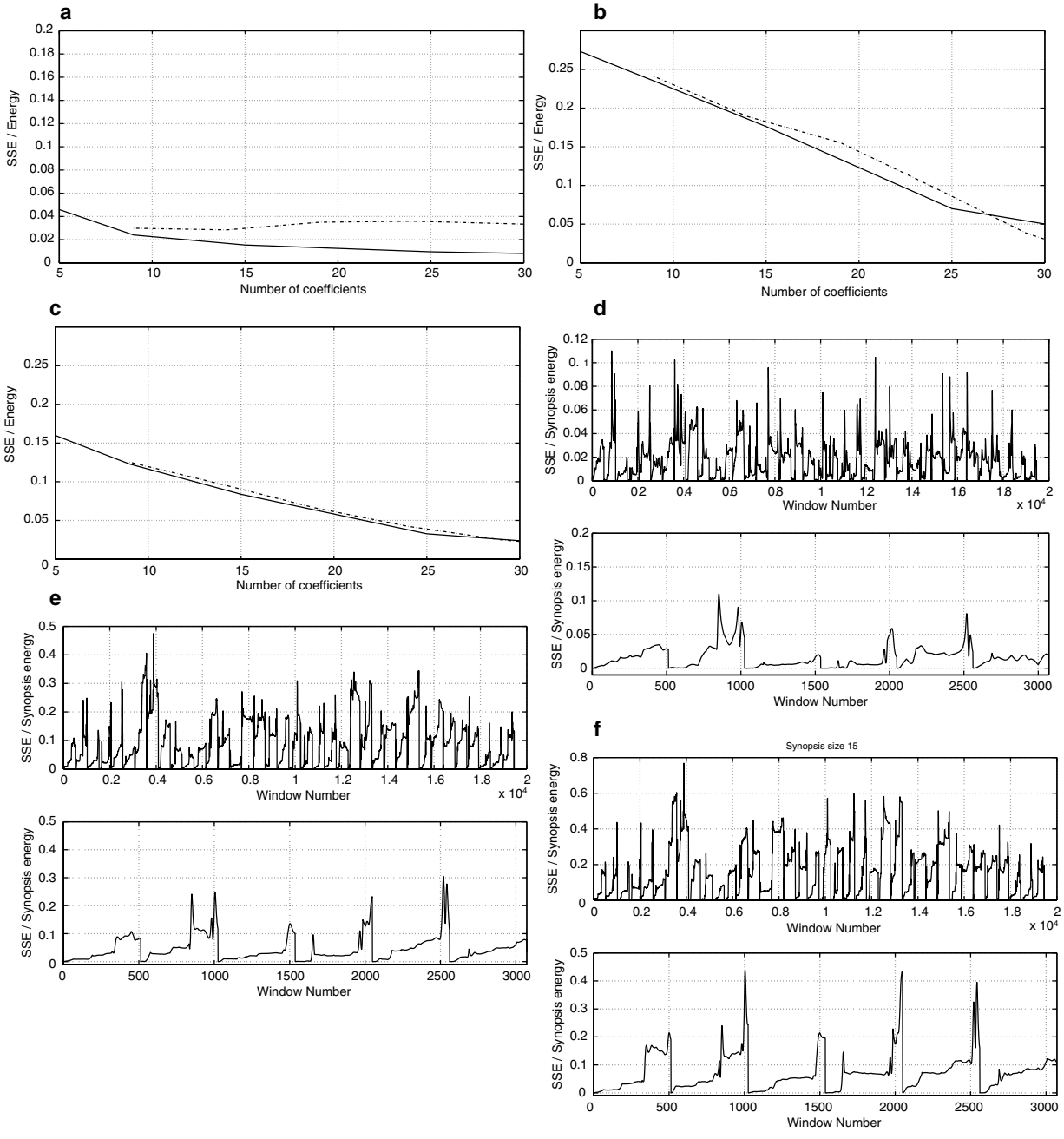

Fig. 6 **a**–**c** Ratio of SSE (between synopsis and original data) and energy of original data for different windows (— exact synopsis, -·- approximate synopsis). **d**–**f** Ratio of SSE between exact and approximate summaries and energy of exact synopsis for varying synopsis sizes. Note that in **a**–**c** we analyze the variation of the ratio with respect to the synopsis size at a certain time instant. On the other hand, in **d**–**f** we analyze the variation of the ratio with time for a constant synopsis size. **a** 5,000th window. **b** 11,000th window. **c** 14,000th window. **d** Synopsis size 25. **e** Synopsis size 20. **f** Synopsis size 15

5.2 Query processing

The results of the first set of experiments suggest that the proposed techniques should achieve satisfactory query performance. In this part, we validate this expectation using real data sets. For the first experiments, we used time series data of size $1{,}080 \times 32{,}500$, which is also used in the first experiments. The window size is selected as 128, and the synopsis size is set to 9, since it was observed that more than 98% of energy is preserved by the approximate synopsis of this size. Particularly, we sought answers to the following questions:

- Point query: Is a particular point in the data set?
- k–NN search: What are the closest k vectors to the query?
- Range query: What are the vectors within the ϵ neighborhood of the query?

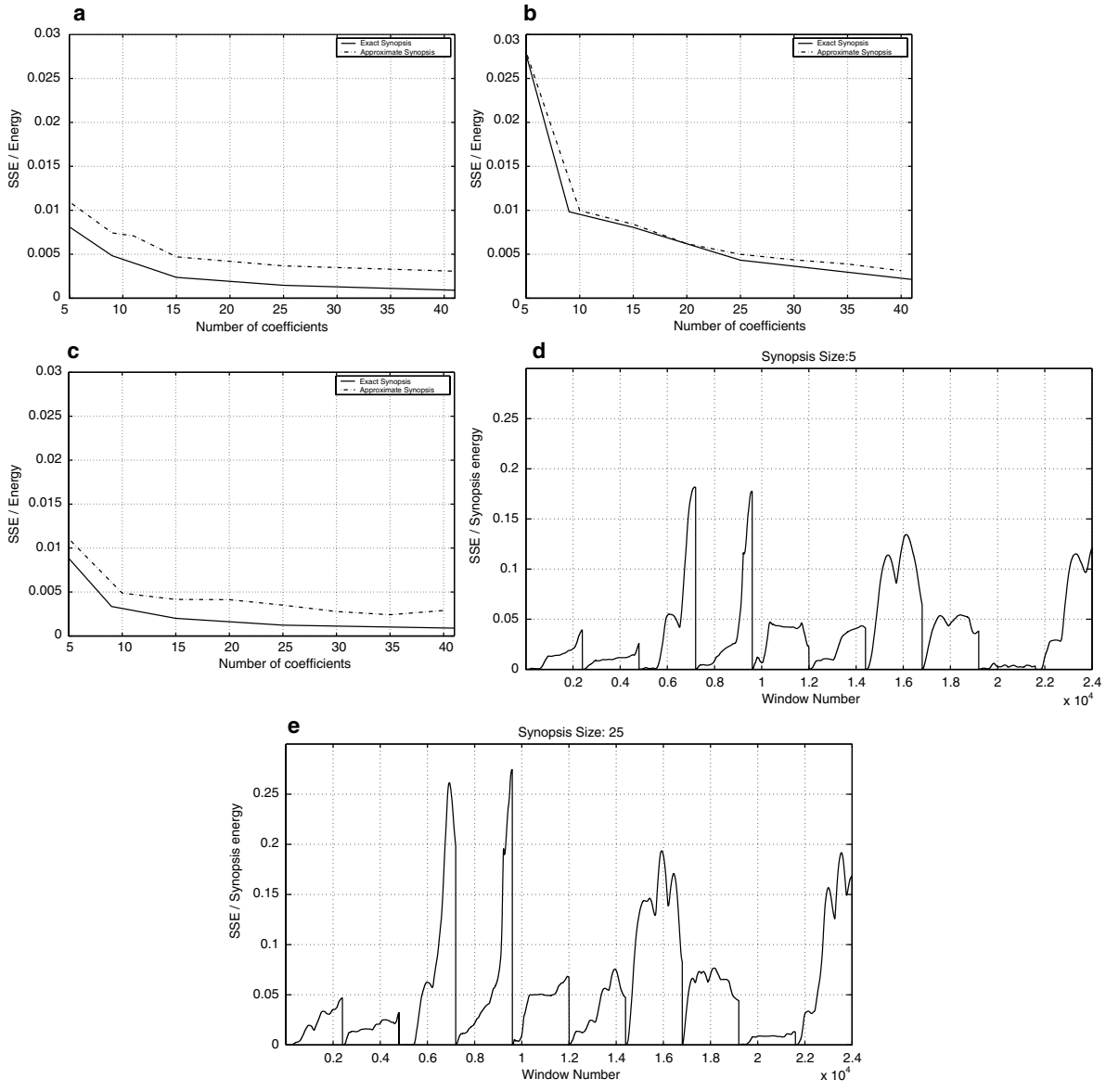

Fig. 7 **a**–bf c Ratio of SSE (between the synopsis and original data) and energy of original data for different windows. **d–e** Ratio of SSE (between exact and approximate summaries) and energy of exact synopsis for varying synopsis sizes. **a** 5,000th window. **b** 11,000th window. **c** 14,000th window. **d** Synopsis size 5. **e** Synopsis size 25

These questions are answered for both continuous and streaming queries. Like the DFT-based summaries of static databases, the proposed techniques underestimate the distance of the query to the data set. Hence we may encounter *false hits* (or *false alarms*), but there are no *false dismissals*, as we show in the experiments.

5.2.1 Continuous query

Unlike one-time queries, continuous queries are issued once and then they run continuously over the database [5]. To evaluate the performance of the algorithm for continuous queries, a query is selected from a similar data set and kept constant as it slides over the data stream. Whenever a new data element is received, the synopsis is updated using our algorithm. Then, the queries are answered using this synopsis. In this experiment, we ran 500 different queries and averaged the results.

We started the experiments by verifying that the synopsis always generated correct answers to point queries. After that, we evaluated the performance of the algorithm for k–NN search. For this purpose, we first found the k-nearest neighbors of the query in the data set using the exact and approximate summaries. Then, we compared the results with the actual k–NN and determined the number of false hits. We repeated this experiment for various different queries and found the average number of false hits as shown in Fig. 8. In this figure, the x-axis denotes the sequence of the

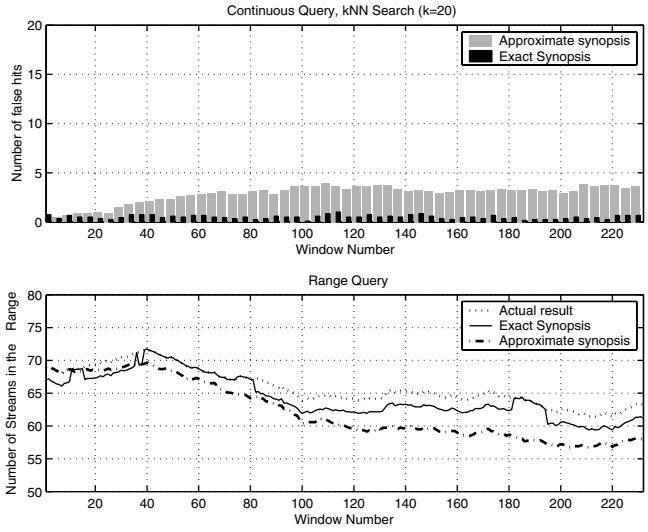

Fig. 8 Variation of number of false hits for a continuous query as the window slides over the stream

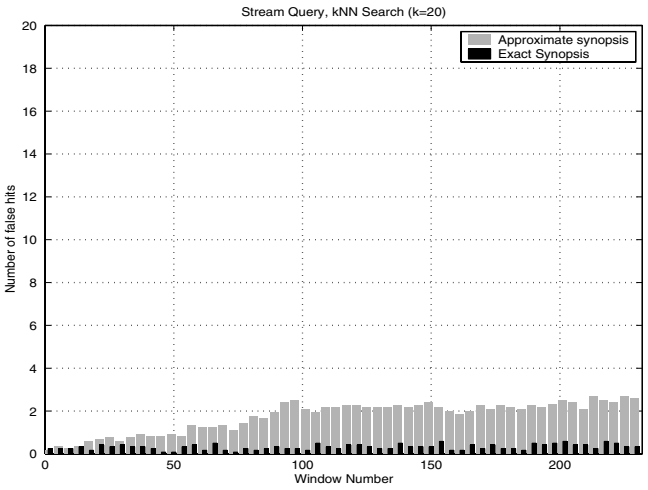

Fig. 9 Variation of number of false hits for streaming query as the window slides over the stream

windows obtained as new elements are received. The corresponding value on the y-axis is the number of false hits computed for that window for $k = 20$. It is seen that the approximate synopsis produces 2–3 more false hits on average. However, the result is still below 5 false hits for 20 nearest neighbors.

The performance is also evaluated based on range queries. A continuous query is defined as for the previous case, and the data points within the ϵ-neighborhood in the latest window of the data stream are searched. For this experiment we selected ϵ as 1% of the average expected distance of the point in the stream. As in the previous experiment, we also found the actual points in the stream that are within the desired distance using the original data and evaluated the performance. The result is shown in Fig. 8. The dotted line shows the actual number of points in the desired neighborhood, while the solid and dash-dotted lines show the number of points selected correctly using the exact and approximate synopsis, respectively.

5.2.2 Streaming query

In some emerging applications such as data recharging and Web monitoring, streaming queries may be encountered as well as streaming data [9]. In this part, we selected a query from a similar stock market data and used this query continuously on the data streams. In addition to the data, the query was also assumed to be a stream in this case. As the window slid over both the stream and the query, we computed the approximate answers based on the synopsis maintained by the proposed method. We also found the exact answers and evaluated the performance by the methods used for continuous queries. Both exact and approximate synopses generated correct answers to point queries. The results for k–NN query with k set to 20 are given in Fig. 9. This plot shows the number of false hits obtained for each window. Out of a total of 232 windows analyzed in this experiment, the average number of false hits caused by the exact synopsis is less than 1. The approximate synopsis causes 2 false hits on average. We can also observe the increase in the number of false hits caused by the approximate synopsis as time progresses. This increase is due to the accumulation of the error, which was analyzed in the first set of experiments (Fig. 5). We also evaluated the proposed method for range queries. A streaming query is defined as for the previous case, and the points within ϵ-neighborhood in the latest window of the incoming query are sought. The results are not included since they are similar to those obtained for continuous queries.

6 Related work

Data stream management systems have very recently attracted the attention of the database and networking communities [5, 7, 9, 10, 11, 14, 16, 18, 20, 21, 23, 25, 27, 36, 46, 49]. In [14], the authors provide an extensive survey of the emerging issues and existing work on data streams. A language for continuous queries over data streams is defined in [5]. A system capable of processing old queries on the new data or new queries on the old data is introduced in [9]. The authors focus on general select project join views and simple classes of aggregates. One of the main application areas of continuous and streaming queries is sensor networks. The advances in sensor technologies have led to the development of sensor networks providing large amounts of data streams. Unlike traditional data sources, sensors suffer from limited power source, processing power, and storage space [10, 36, 49]. In [36], the authors present an architecture to manage multiple queries over many sensor inputs. Processing queries in sensor networks is discussed in [49]. The authors develop a query layer and analyze in-network aggregation, the interaction of the in-network aggregation with the wireless routing protocol, and distributed query processing. In [18], the authors concentrate on monitoring

continuous data streams and introduce a system to monitor streams from sensors.

Motwani et al. [16, 20] discuss maintaining statistics over sliding windows. The authors present methods to approximate the variance, k-medians clustering, and simple statistics over sliding windows with proven error bounds. Histogram-based methods are widely used for selectivity estimation. Maintenance of classical partition-based histograms are discussed in [26]. To increase the accuracy of the classical histogram-based methods, transform-based histogram methods are introduced [38]. The maintenance of such histograms, which are relatively more difficult than classical histograms, is discussed in [34, 38]. In [38], the authors develop a method based on probabilistic counting and sampling to dynamically maintain wavelet-based histograms. Gehrke et al. [25] present a one-pass technique to answer correlating aggregate queries over streams using histograms summaries. In [34], the authors use DCT-based compressed histograms for multidimensional selectivity estimation. The changes in the histogram are reflected in the summary using the linearity property of the DCT.

Generating online summaries of data streams based on wavelet transform approximations is discussed in [27]. In this paper, the authors discuss generating one pass summaries of data streams. However, our technique differs from their technique in many aspects. They use a sketch, using a random projection approach, e.g., the one from [3], of the underlying data set that can be used to estimate the wavelet coefficients. In their technique, the sketch of the data, not the wavelet coefficients, is updated incrementally as new data points are received. That is, while the update process of the sketch is online, computation of the wavelet-based synopsis is a batch process, which runs on a sketch of data corresponding to a certain period, e.g., one day. As a result, obtaining a wavelet-based synopsis requires extra processing ($\Omega(N \log N)$) on top of online maintenance of the sketch with logarithmic time and space. Although this cost can be reduced (as is done in their final experiment) by computing 1 coefficient for every 1,000 data items in the background, this degrades the accuracy, as is pointed out by the authors [27]. A more serious limitation is that the background computation catches up with the batch computation, and even surpasses it, if more and more wavelet coefficients are computed.

In contrast, our technique computes the synopsis incrementally in $O(M)$ time and space ($M \ll N$, is the synopsis size), although the focus of our work has been on sliding window computations. Furthermore, it produces the exact set of transform coefficients (identical to computing the transform offline and selecting the best coefficients), unlike the technique in [27]. As we have shown, our technique maintains the synopsis of the sliding window covering most recent N points successfully. It can produce both exact synopses (with $O(M)$ time and $O(N)$ space) and approximate synopses (with $O(M)$ time and $O(N)$ space).

On the other hand, the technique in [27] is extended to concatenated streams by defining the λ-aging data stream. This approach can be used to compute the synopsis of a sliding window over a block of data. Decreasing the size of the block to a single data point would make this process truly incremental, as in our case. However, this is prohibitive for their technique, since implementing this would require $\Omega(N \log N)$ time to compute the wavelet coefficients from the sketch as discussed above. In [27], it is also argued that the sketch can directly be used to answer queries online, which could have been an alternative to our solution. However, this produces the worst approximations as pointed out in [27].

Finally, computation of a moving Fourier transform has been discussed in the signal processing area [2, 40], and DFT over sliding windows has been studied in [12].

7 Conclusions and future work

Summaries of large databases have been used extensively in many applications. Although time series data are dynamic by nature, the current summary generation techniques have been mostly for static time series databases. Due to the recent applications, in which dynamic time series data are collected from many sources, online and update-efficient methods are needed to generate and maintain the summaries. In this paper, we develop a transformation-based framework to online summary generation for large-scale and dynamic time series data. Specifically, we concentrate on DFT-based synopsis generation and introduce a recursive method to update the highest energy transform coefficients of the time series data. The computational complexity of the proposed technique depends on the size of the synopsis, while the computational complexity of DFT is $N \log N$. Furthermore, we develop a one-pass technique to compute an approximate synopsis with proven error bound using $O(M)$ space as well as $O(M)$ time. In Sect. 5, we demonstrate our results using four real data sets and evaluate the performance using continuous and stream queries over dynamic time series data.

A possible extension of the work is to monitor the performance of the summaries. If the performance is below a threshold, the size of the synopsis can be increased adaptively. Also, for cases when the frequency distribution of the data has significant changes, the set of coefficients that are updated can be revised.

Acknowledgements This material was prepared with the support of the U.S. Department of Energy (DOE) Award No. DE-FG02-03ER25573. However, any opinions, findings, conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect the views of DOE.

We would like to acknowledge the constructive comments made by the reviewers, which have improved the presentation of the paper significantly.

This work was completed when the first author was with the Department of Computer Science and Engineering, The Ohio State University

Appendix: Calculation of A and P

The DFT matrix is given as

$$T_{ij} = W_N^{-(i-1)(j-1)}, \quad i, j = 1, \ldots, N\,.$$

During the update process of the synopsis, H is modified as follows: the rows from 2 to n are shifted up, while the first row comes to the bottom. If we apply the same operations to T, the resulting matrix becomes

$$\acute{T}_{ij} = \left\{ \begin{array}{ll} W_N^{-i(j-1)} & \text{for } i = 1, \ldots, N-1 \\ 1 & \text{for } i = N \end{array} \right\}\,.$$

We are looking for a matrix A such that $\acute{T} = T \times A^{-1}$. Note that, if we multiply each column of T by $W_N^{-(j-1)}$, we can obtain $\acute{T}$. The result for rows $i = 1, \ldots, n-1$ are clear. The result for the last row stems from the fact that DFT coefficients are periodic with 2π. As a result, A^{-1} is found as shown in Eq. 7.

P is given by Eq. 5 as $P = (H^T H)^{-1}$. As a result, we obtain P^{-1} as

$$P^{-1} = [h_1^T h_2^T \cdots h_N^T] \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_N \end{bmatrix},$$

$$P^{-1} = \sum_{i=1}^{N} h_i^T h_i\,.$$

Hence, interchanging the rows does not change P. Remember that h_i is given by

$$h_i = \big[1\, W_N^{-(i-1)} W_N^{-2(i-1)} \cdots W_N^{-\frac{M}{2}(i-1)} W_N^{-(N-\frac{M}{2})(i-1)} \cdots W_N^{(N-2)(i-1)} W_N^{(N-1)(i-1)}\big]/N\,. \tag{A1}$$

As a result, the diagonal of the product $h_i^T h_i$ is $\frac{1}{N^2}$, while the other entries are of the form $W_N^{m(i-1)}$, where m takes the values $\mp 1, \mp 2, \ldots, \mp(N-1)$:

$$\sum_{i=1}^{N} W_N^{m(i-1)} = \sum_{i=1}^{N} e^{-j\frac{2pi}{N}m(i-1)} = \frac{1 - e^{j\frac{2\pi mN}{N}}}{1 - e^{j\frac{2\pi m}{N}}} = 0\,. \tag{A2}$$

since $m \neq 0$. Thus,

$$P^{-1} = \sum_{i=1}^{N} h_i^T h_i = \frac{1}{N} I_{k\times k}\,.$$

References

1. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms (1993)
2. Albrecht, S., Cumming, I., Dudas, J.: The momentary fourier transformation derived from recursive matrix transformations. In: Proceedings of the 13th International Conference on Digital Signal Processing (1997)
3. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. In: ACM STOC (1996)
4. Ayad, A.M., Naughton, J.F.: Static optimization of conjunctive queries with sliding windows over infinite streams. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (2004)
5. Babu, S., Widom, J.: Continuous queries over data streams. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (2001)
6. Berchtold, S., Bohm, C., Kriegel, H.-P.: The Pyramid-Technique: Towards breaking the curse of dimensionality. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (1998)
7. Bulut, A., Singh, A.: Swat: Hierarchical stream summarization in large networks. In: Proceedings of the International Conference on Data Engineering (2003)
8. Castleman, K.R.: Digital Image Processing. Englewood Cliffs: Prentice-Hall (1996)
9. Chandrasekaran, S., Franklin, M.J.: Streaming queries over streaming data. In: Proceedings of the International Conference on Very Large Data Bases (2002)
10. COUGAR. The cougar sensor database project: the network is the database. http://www.cs.cornell.edu/database/cougar/index.htm/
11. Dobra, A., Garofalakis, M., Gehrke, J.E., Rastogi, R.: Processing complex aggregate queries over data streams. In: ACM SIGMOD (2002)
12. Douglas, S.C., Soh, J.K.: A numerically-stable slidingwindow estimator and its application to adaptive filters. In: Proceedings of the 31st Asilomar Conference on Signals, Systems, and Computers (1997)
13. Egecioglu, O., Ferhatosmanoglu, H., Ogras, U.: Dimensionality reduction and similarity computation using inner product approximations. IEEE Trans. Knowl. Data Eng. **16**(6), 714–726 (2004)
14. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proceedings of the 21st ACM Symposium on Principles of Database Systems (2002)
15. Babcock, B., Babu, S., Datar, M., Motwani, R.: Chain: Operator scheduling for memory minimization in data stream systems. In: Proceedings of the ACM SIGMOD Interantional Conference on Management of Data (2003)
16. Babcock, B., Datar, M., Motwani, R., O'Callaghan, L.: Sliding window computations over data streams. In: Proceedings of the Symposium on Principles of Databases Systems (2003)
17. Abadi, D.J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: A new model and architecture for data stream management. In: Proceedings of International Conference on Very Large Data Bases (2003)
18. Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., Zdonik, S.: Monitoring streams – a new class of DBMS applications. In: International Conference on Very Large Data Bases (2002)
19. Chakrabarti, K., Garofalakis, M., Rastogi, R., Shim, K.: Approximate query processing using wavelets. In: Proceedings of the International Conference on Very Large Data Bases (2000)

20. Datar, M., Gionis, A., Indyk, P., Motwani, R.: Maintaining stream statistics over sliding windows. In: Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (2002)
21. Motwani, R., Widom, J., Arasu, A., Babcock, B., Babu, S., Datar, M., Manku, G., Olston, C., Rosenstein, J., Varma, R.: Query processing, approximation, and resource management in a data stream management system. In: Proceedings of the CIDR Conference (2003)
22. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast subsequence matching in time-series databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (1994)
23. Gao, L., Wang, X.: Continually evaluating similaritybased pattern queries on a streaming time series. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (2002)
24. Garofalakis, M., Gibbons, P.B.: Wavelet synopses with error guarantees. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (2002)
25. Gehrke, J., Korn, F., Srivastava, D.: On computing correlated aggregates over continual data streams. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (2001)
26. Gibbons, P.B., Matias, Y., Poosala, V.: Fast incremental maintenance of approximate histograms. In: Proceedings of the Internatinal Conference on Very Large Data Bases (1997)
27. Gilbert, A., Kotidis, Y., Muthukrishnan, S., Straus, M.: Surfing wavelets on streams: one pass summaries for approximate aggregate queries. In: International Conference on Very Large Data Bases (2001)
28. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proceedings of the International Conference on Very Large Data Bases (1999)
29. Kailath, T.: Modern Signal Processing. Berlin, Heidelberg, New York: Springer (1985)
30. Kang, J., Naughton, J.F., Viglas, S.: Evaluating window joins over unbounded streams. In: Proceedings of the International Conference on Data Engineering (2003)
31. Kanth, K.V.R., Agrawal, D., Singh, A.: Dimensionality reduction for similarity searching in dynamic databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (1998)
32. Karhunen, H.: Uber lineare methoden in der wahrscheinlichkeitsrechnung. Ann. Acad. Sci. Fennicae, Ser. A1 Math.-Phys. **37**, 3–79 (1947)
33. Keogh, E.J., Chakrabarti, K., Mehrotra, S., Pazzani, M.J.: Locally adaptive dimensionality reduction for indexing large time series databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (2001)
34. Lee, J., Kim, D., Chung, C.: Multi-dimensional selectivity estimation using compressed histogram information. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (1999)
35. Loeve, M.: Fonctions aleatoires de seconde ordre. Processus Stochastiques et Mouvement Brownien. Paris: Hermann (1948)
36. Madden, S., Franklin, M.J.: Fjording the stream: an architecture for queries over streaming sensor data. In: Proceedings of the International Conference on Data Engineering (2002)
37. Matias, Y., Vitter, J.S., Wang, M.: Wavelet based histograms for selectivity estimation. In: Proceedings of the ACM Sigmod International Conference on Management of Data (1998)
38. Matias, Y., Vitter, J.S., Wang, M.: Dynamic maintenance of wavelet-based histograms. In: International Conference on Very Large Data Bases (2000)
39. Mendel, J.: Lessons in Estimation Theory for Signal Processing, Communications, and Control. Englewood Cliffs: Prentice-Hall (1995)
40. Populis, A.: Signal Analysis. New York: McGraw-Hill (1977)
41. Rafiei, D., Mendelzon, A.: Similarity-based queries for time series data. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (1997)
42. Rafiei, D., Mendelzon, A.: Efficient retrieval of similar time sequences using dft. In: Proceedings of the International Conference on Foundations of Data Organization and Algorithms (FODO) (1998)
43. Rao, K.R., Yip, P.C.: The Transform and Data Compression Handbook. Boca Raton: CRC (2001)
44. Seidl, T., Kriegel, H.P.: Optimal multi-step k-nearest neighbor search. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. Chicago: ACM (1998)
45. Shumway, R.H., Stoffer, D.S.: Time Series Analysis and Its Applications. Berlin, Heidelberg, New York: Springer (2000)
46. Viglas, S., Naughton, J.F.: Rate-based query optimization for streaming information sources. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Madison, WI (2002)
47. Vitter, J.S., Wang, M.: Approximate computation of multidimensional aggregates of sparse data using wavelets. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (1999)
48. Wu, D., Agrawal, D., El Abbadi, A., Smith, T.R.: Efficient retrieval for browsing large image databases. In: Proceedings of the Conference on Information and Knowledge Management, pp. 11–18 (1996)
49. Yao, Y., Gehrke, J.: Query processing for sensor networks. In: Proceedings of CIDR (2002)