Incremental Maintenance of Online Summaries[1] Over Multiple Streams

Fatih Altiparmak[1], Ertem Tuncel[2], Hakan Ferhatosmanoglu[1]

[1]Department of Computer Science and Engineering, The Ohio State University

[2]Department of Electrical Engineering, University of California, Riverside

emails : ertem@ee.ucr.edu, {altiparm, hakan}@cse.ohio-state.edu

Abstract

We propose a novel approach based on predictive quantization (PQ) for online summarization of multiple time-varying data streams. A synopsis over a sliding window of most recent entries is computed in one pass and dynamically updated in constant time. The correlation between consecutive data elements is effectively taken into account without the need for preprocessing. We extend PQ to multiple streams and propose structures for real-time summarization and querying of a massive number of streams. Queries on any subsequence of a sliding window over multiple streams are processed in real-time. We examine each component of the proposed approach, prediction and quantization, separately and investigate the space-accuracy tradeoff for synopsis generation. Complementing the theoretical optimality of PQ-based approaches, we show that the proposed technique, even for very short prediction windows, significantly outperforms the current techniques for a wide variety of query types on both synthetic and real data sets.

I. Introduction

Although many solutions have been proposed in the literature for maintaining dynamic databases, such as efficient insertion of new data objects, efficient update of data with evolving sequences has attracted attention only very recently [31]. The elements of data objects change over time in many cases such as in a multiple data stream application where a new element of each data sequence, is periodically inserted to the database. For example, in stock market analysis, a database is usually formed by incrementally storing multiple data streams of stock prices. The current stock value for each company is periodically inserted to the corresponding company's stock profile for real-time analysis of stock price movements. There are about 50,000 securities trading in the United States, and every second up to 100,000 quotes and trades (ticks) are generated [31]. The users are often concerned with finding correlations and trends on a specific stock or a set of stocks.

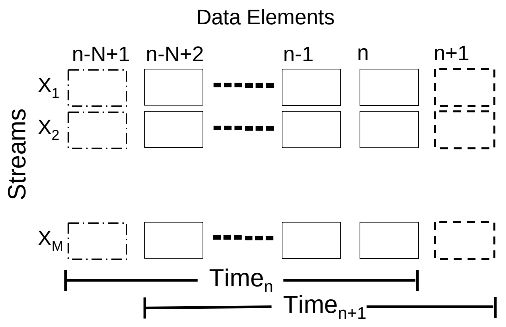


Fig. 1. Data under the Sliding Windows at Time$_n$ and Time$_{n+1}$

The data streams are usually sized by a window on the most recent data, e.g., the last 30 days, or the last 24 hours, etc. This is called the sliding window model. In this model, data elements arrive in a stream and only the last N (window size) elements to have arrived are considered relevant at any moment. For multiple (M) streams, the model is summarized in Figure 1. At each time instant, a new data element arrives and the oldest entry in the window is forgotten for each stream. In order to be able to respond to the volatility of the stock market, the user requests need to be analyzed continuously whenever new stock prices arrive. Besides the stock market applications, several other applications, such as telecommunications data management, intrusion detection, and sensor networks, involve periodically querying a database of multiple streams. Online and ad-hoc queries are continuously executed to discover useful patterns over data. Immediate responses are desirable since these applications are usually time-critical and important decisions need to be made upon the query results. To address these issues and several others, the concept of *data stream management systems* (DSMS) has recently attracted the attention of the database community [4], [7], [9], [14], [15], [19], [20], [27], [28], [29].

A fundamental challenge in data stream management is to develop an online technique to summarize multiple data streams. A general-purpose summary can be utilized by a large number of data mining and management tasks over multiple streams, such as clustering, classification, change detection, statistical monitoring, selectivity estimation, query optimization, and query processing. While reducing the size of the data, the resulting summaries are typically designed to have certain qualities needed for the applications, such as preserving the original pairwise distances as much as possible [1], [18].

Most of the stream data of practical importance, such as stock values, telecommunications records, and sensor data, possess a high amount of correlation between nearby sample values. Consecutive elements of a stream are highly correlated to each other. Therefore, there is an inherent redundancy in the representation of the data, and efficient data summarization techniques rely on removal of this redundancy to reduce the size of the summary. Recently, several signal transformation techniques have been proposed to reduce redundancy in data streams [6], [12], [24], [25], [31].

These techniques, as their traditional analogs, store the most significant *transform coefficients* of the time series signal, and discard the rest of the coefficients, after applying one of Principal Component Analysis (PCA), Discrete Fourier Transform (DFT), Discrete Cosine

Transform (DCT), or Discrete Wavelet Transform (DWT). For example, Statstream [31] is a stream monitoring system that utilizes a DFT-based summary. The DFT of the stream data is computed and a summary is continuously generated by taking the top coefficients. Similarly, Surfing-Wavelets [12], Awsom [24] and Stardust [6] utilize wavelets for data summarization. Spirit [25] employs the first K principal components of the PCA for monitoring multiple streams. The principal components are updated at each time instant in $O(KM)$ time, where K is the number of corresponding principal components of that time instant and M is the number of streams.

As a powerful alternative to signal transforms, one can consider the scalar quantization approaches that have been successful in traditional high dimensional databases [5], [26], [30]. In the VA-file method [30], the data is summarized by quantizing each dimension independently, and mapping each data point into the bit approximation of the cell that contains it. This approach is especially attractive for data stream summarization, because each newly arrived element can be quantized without affecting the previously quantized dimensions, i.e. time instants both in sliding window and semi-infinite window cases. This results in fast and one-pass updates. However, the performance of such an approach heavily suffers from the fact that correlation between the data dimensions are completely ignored. Removal of the redundancy via signal transforms followed by quantization is more effective than purely quantization-based or purely transform-based approaches [11]. Following such an intuition, VA$^+$-file [10] has been proposed for highly correlated data (such as time series). In this approach, first, the dimensions of the data are decorrelated via the application of a transform, then the available bits specified by the size of the synopsis are allocated non-uniformly among the dimensions. Finally, a scalar quantizer is designed for each transformed dimension independently. Unfortunately, although shown to be effective for static databases, applying such preprocessing is infeasible for the case of streaming data because of real-time requirements.

All these observations call for an algorithm for summarizing multiple data streams that would enjoy the best of both worlds, i.e., that would be as effective as transform domain processing in removing the inter-dimensional redundancy and as fast as scalar quantization for online computation of the summary. In this paper, we propose such an algorithm based on a differential representation and predictive quantization of the data. Prediction has been applied in signal processing and multimedia applications for transmission of a *single* data stream of data, such as video or speech signal. A prediction model is used over a long period so that the cost of storing the coefficient of this setting is amortized. In our approach, which we call PQ-Stream, the correlation between the data elements is exploited through the prediction of the incoming elements of *multiple* streams in terms of the latest few elements. In this prediction, the error is efficiently quantized at each instant independently by utilizing a different setting for that instant. We find the optimum parameters for the setting immediately over multiple streams and with the prediction error we also save the parameters of the setting for each instant to reconstruct the previous data points.

Our contributions in this paper are as follows:

- We adopt predictive quantization which is theoretically shown to achieve the performance of any transform-based data summarization technique in terms of representation quality for a single signal. Although very powerful, to the best of our knowledge, predictive summarization has not been utilized before for data stream summarization.
- We extend PQ to *multiple* streams and propose techniques to summarize and query massive amounts of streams in real time.

- The summary over a sliding window of most recent entries is computed in a one-pass fashion. The correlation between consecutive data elements are effectively taken into account without the need of any preprocessing.
- The running time of our synopsis update algorithm is *independent* of the size of the sliding window, N.
- The synopsis saved for each stream gives *flexibility* to run queries on *any subsequence* of the latest N data elements, where N is the size of the sliding window.
- Supporting the theoretical effectiveness of PQ-Stream, our experiments, performed on both real and synthetic data sets, demonstrate its superiority over the current techniques, in terms of both quality of data representation query results.

The rest of the paper is organized as follows. Section II gives the technical motivation of the proposed approach. Section III outlines the PQ-Stream scheme, by describing the generation and update of the synopsis, as well as query processing over the synopsis. Section IV presents an extensive performance evaluation of PQ-Stream including comparisons with the current techniques using various queries. Section V discusses the current synopsis generation techniques. Finally, Section VI concludes the paper.

II. Technical Background and Motivation

In this section, we provide a background on transform coding and quantization, and present the technical motivation and basis of the proposed approach. The notation and symbols used throughout the paper are described in Table 1.

A. Transform Coding

The most common approach for summarization of high dimensional data has been to apply transformation techniques such as DFT, DCT, and DWT and utilizing the most significant

Symbol	Meaning
$x_i[j]$	*j^{th} data point of the i^{th} stream*
N	*Size of the sliding window*
M	*Total number of streams*
K	*Size of the prediction window*
$p_k[n]$	*k^{th} prediction coefficient of time-instant n*
$\tilde{x}_i[j]$	*Estimation using previous K elements*
$\hat{x}_i[j]$	*Reconstructed data*
$e_i[j]$	*Prediction Error*
$\hat{e}_i[j]$	*Quantized Prediction Error*
Q_n	*Quantizer for time-instant n*
f_n	*Encoder for time-instant n*
g_n	*Decoder for time-instant n*

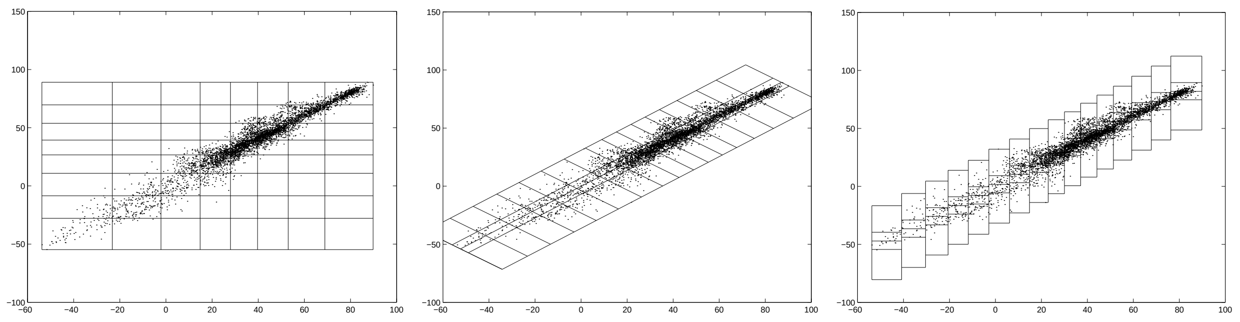
TABLE I

NOTATION AND SYMBOLS

dimensions of the transformed data. Recent stream summarization techniques also adapted a similar approach and used DFT [31] and DWT [6], [12], [24] to summarize the data for indexing, query processing, and mining of data streams. The optimal approach, in the sense of minimizing the quality degradation, is to transform the data using the Karhunen-Loeve Transform (KLT) [11, Section 8.6], which also appears in the literature as Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). Because of being dependent on the whole data set and its high computational cost, KLT has not been popularly used to summarize large and highly dynamic databases.

Orthogonal to the transform based approach, scalar quantization has also been proposed for high dimensional data summarization. The most prominent example of this approach, where the dimensions are independently quantized, is the VA-file method [30]. Scalar quantization is a natural choice also for data streams, since each newly arrived data element can be quantized without affecting the previously quantized data. However, it fails to exploit the high correlation between data dimensions, which is crucial in data stream summarization.

It is well-established that the optimal data summarization scheme is the combination of the



(a) Plain quantization (b) Quantization of KLT coefficients (c) Predictive quantization

Fig. 2. Comparison of the resultant quantization cells.

transformation and quantization approaches (which is referred to as *transform coding* in the data compression literature), i.e., distributing a total quota of bits over transform coefficients (see [11, Section 8.3]), as opposed to simply taking the most significant coefficients with full precision (quantization with 32 bits/coefficient), and discarding the rest of the coefficients (quantization with 0 bits/coefficient). This two step process is successfully applied to static databases [10], where long processing of data may be amortized with gains in query processing. However, it is not of immediate use as an online technique, because such preprocessing is infeasible for the case of streaming data.

B. Predictive Quantization

Prediction has been employed in applications such as multimedia and speech coding, where the data can be represented tailored to the underlying signal. To the best of our knowledge, prediction [21] and quantization [22] have been applied to data streams and databases separately, however, predictive quantization has not been used before. A prediction function f estimates the value of the sample $x[n]$, for time instant n, using previous samples, i.e.,

$$\tilde{x}[n] = f(x[n-1], x[n-2], \ldots, x[n-N+1]).$$

It is only the prediction error $e[n] = x[n] - \tilde{x}[n]$, which usually has much narrower dynamic range compared to $x[n]$, that is to be efficiently represented. Samples of the prediction error can be theoretically shown to be uncorrelated with the right choice of the prediction function $f(\cdot)$, which indicates the success of the PQ approach in terms of removing the redundancy between data elements. The simplest type of prediction, which is performed by taking a linear combination of previous $K \ll N$ samples, i.e.,

$$\tilde{x}[n] = \sum_{k=1}^{K-1} p_k x[n-k]$$

is called *linear prediction*. We adopt in this paper this kind of prediction because of its tractability and ease of use.

An important question is how PQ compares in quality with a purely transform domain processing method, a purely scalar quantization based method, and finally the transform coding method, which performs scalar quantization over transform coefficients. As we will discuss next, PQ achieves as high a performance as transform coding in terms of reconstruction quality. Since transform coding is the optimal approach, this in turn implies the superiority of PQ over purely transform domain processing and purely scalar quantization based methods. We will experimentally demonstrate the validity of this claim, even when PQ is applied over multiple streams. A major advantage of using PQ instead of transforms is that in sliding window applications, where only a synopsis of the most recent data is stored in main memory, it is very difficult to update the synopsis if it is based on keeping the transform coefficients. On the other hand, in a scheme based on predictive quantization, the whole synopsis can be updated in constant time (in terms of the sliding window length, N) both in semi-infinite and sliding window cases, as will be shown later.

Before going into technical details of the optimality of PQ-based methods, however, let us provide some intuition on a real-world example. Figure 2 exhibits a practical comparison of scalar quantization, transform coding, and PQ. The collection of points in the 2-D plane represents two consecutive samples $x[n-1]$ and $x[n]$ taken from thousands of different streams. Independent quantization of $x[n-1]$ and $x[n]$, as shown in Figure 2(a) results in a poor utilization of all the available 64 quantization cells, which can be represented by 6 bits. In transform coding, the points are first transformed, i.e., rotated, and then each transform coefficient is quantized separately. In the PQ scheme, on the other hand, first $x[n-1]$ is quantized independently, and then its *reconstructed*, or *decompressed* version $\hat{x}[n-1]$ is used for the estimation of $x[n]$:

$$\tilde{x}[n] = p_1 \hat{x}[n-1] \ .$$

The prediction error $e[n] = x[n] - \tilde{x}[n]$ is then quantized. In Figure 2(b) and (c), the equivalent number of quantization cells are depicted on the original signal domain. In Figure 2(b), 4 bits are used to quantize one of the resulting dimensions after the transformation and the remaining 2 bits are used for the other one. Where as in Figure 2(c), 4 bits are used to quantize $x[n-1]$ and the remaining 2 are used for the error. It can be observed that both methods are superior to plain quantization, and comparably efficient in covering the signal points using 64 cells.

C. Theoretical Optimality of PQ

For a data sequence of length N, the *coding gain* of a transform is defined as

$$G_{\text{tc}} = \frac{D_{\text{direct}}}{D_{\text{tc}}}$$

where D_{direct} is the resultant mean square error (MSE) of the scheme where signal samples are quantized without any transformation, and D_{tc} is the resultant MSE of the particular transformation followed by bit allocation and quantization. In the idealized case where the signal samples are jointly Gaussian, the coding gain of KLT is given by

$$G_{\text{KLT}} = \frac{\left(\prod_{n=1}^{N} \sigma_n^2\right)^{1/N}}{\left(\prod_{n=1}^{N} \lambda_n^2\right)^{1/N}}$$

where σ_n^2 and λ_n^2 are the variances of original signal samples and of KLT coefficients, respectively. For *stationary* processes, where the statistics do not change with time, σ_n^2 becomes a constant, and therefore,

$$G_{\text{KLT}} = \frac{\sigma_n^2}{\left(\prod_{n=1}^{N} \lambda_n^2\right)^{1/N}} \ . \tag{1}$$

The coding gain of prediction is defined similar to that of transform coding:

$$G_{\text{p}} = \frac{D_{\text{direct}}}{D_{\text{p}}}$$

where D_{p} is the resultant MSE after prediction and quantization. For stationary processes, it can be shown [11] that

$$G_{\text{p}} = \frac{\sigma_n^2}{\mu_n^2} \tag{2}$$

where μ_n^2 is the variance of the prediction error $e[n]$. Therefore, it is the quality of the prediction, in the sense of minimizing the variance of the prediction error, that determines the resultant coding gain.

The premise of our approach is the following theorem (Theorem 4.9.3 in [11]).

Theorem 1: For a stationary Gaussian process, as the prediction window length, K, and the sliding window length, N, become very large,

$$\mu_n^2 \longrightarrow \left(\prod_{n=1}^{N} \lambda_n^2\right)^{1/N}$$

This asymptotic result, together with (1) and (2), suggests that the performance of predictive quantization with a large window approaches that of the optimal transform coding. In practice, the quality of prediction quickly converges to its maximum as K increases, hence a much smaller prediction window of length $K \ll N$ is observed to be as effective as infinite K. The prediction is typically applied over a single signal and through the estimation of the time-statistics of the data. Compression can be achieved by using the same set of prediction coefficients $\{p_k\}$ over a long enough period so that the cost of storing those coefficients in addition to the quantized prediction error is amortized. In the current problem, however, we have multiple streams $\{x_1[n], \ldots, x_M[n]\}$, which can statistically exhibit a highly non-stationary behavior. Hence, we adopt a different approach and estimate new prediction coefficients for each time instant $n = N$, but use the same coefficients to predict the *whole* set of incoming data elements $\{x_1[N], \ldots, x_M[N]\}$ each in terms of the corresponding few previous data elements, i.e.,

$$\tilde{x}_m[n] = \sum_{k=1}^{K-1} p_k[n] x_m[n-k] \ .$$

Hence, we amortize the additional cost of storing $\{p_k\}$ over a large number of streams, instead of over long time periods. Even though Theorem 1 no longer holds for non-stationary and non-Gaussian streams, the theoretical competitiveness of PQ carries over to practice, as we show experimentally, thanks to the feasibility of updating the prediction coefficients at every instant.

III. PQ-Stream Technique

In this section we describe the proposed PQ-Stream technique in detail. We first present our algorithm for online generation of a PQ-based synopsis over multiple streams. We define the elements of the synopsis kept for each instant and present a technique to compute each of them. We then introduce our algorithm to update the synopsis in O(K^2M) time, i.e., O(K^2) amortized time per data element, as the data points arrive. Finally, we describe how the dynamically maintained synopsis is utilized by query processing algorithms. We discuss how to reconstruct actual data from the synopsis and process a variety of query types over the synopsis.

A. Generation of Synopsis over Multiple Streams

Let $x_m[n]$ for $1 \leq m \leq M$ and $0 \leq n \leq N-1$ denote the value of the m^{th} data stream at time instant n. Here, $n = N-1$ refers to the most recent sample, and $n = 0$ is the oldest sample which is to be forgotten when a new sample arrives. Using our summarization technique which will be described below, estimates of $x_m[n-1], \ldots, x_m[n-K]$, denoted by $\hat{x}_m[n-1], \ldots, \hat{x}_m[n-K]$, are to be *reconstructed* using the synopsis of the data, and used for prediction of $x_m[n]$. We adopt the linear prediction scheme

$$\tilde{x}_m[n] = \sum_{k=1}^{K} p_k[n]\hat{x}_m[n-k] \tag{3}$$

for all m, where $p_k[n]$ are called the *prediction coefficients*. Note that the coefficients are themselves functions of time n. However, the additional cost of storing $p_k[n]$ will be negligible compared to storing the synopsis of all M streams.

The prediction coefficients are computed in a least squares manner, i.e., so as to minimize the energy

$$\sum_{m=1}^{M} e_m[n]^2 ,$$

where

$$e_m[n] = x_m[n] - \tilde{x}_m[n] \tag{4}$$

is the *prediction error*. Minimization of this energy will result in a more efficient summarization, simply because the summary is comprised of *quantized* versions of $e_m[n]$. The energy minimizing coefficients can be computed in a standard manner (e.g., see [11, Section 4.3]), i.e., as the solution of the so-called Yule-Walker equations

$$\mathbf{A}[n] \begin{bmatrix} p_1[n] \\ p_2[n] \\ \vdots \\ p_K[n] \end{bmatrix} = \mathbf{c}[n] , \tag{5}$$

where

$$A_{ij}[n] = \sum_{m=1}^{M} \hat{x}_m[n-i]\hat{x}_m[n-j]$$

and

$$c_i[n] = \sum_{m=1}^{M} x_m[n]\hat{x}_m[n-i] .$$

The prediction error $e_m[n]$ is the input to the *quantizer* function Q_n, which is specifically designed for the distribution of $e_m[n]$ along m. That is, Q_n is designed in real time so as to minimize

$$\sum_{m=1}^{M} (e_m[n] - \hat{e}_m[n])^2 \tag{6}$$

where $\hat{e}_m[n]$ is the quantized version of $e_m[n]$. Although intuitively sane, this objective must still be justified. The justification is that the reconstructed data, which we denote by $\hat{x}_m[n]$, will be computed using $\hat{e}_m[n]$ as

$$\hat{x}_m[n] = \tilde{x}_m[n] + \hat{e}_m[n] \tag{7}$$

and therefore using (4), we obtain

$$\begin{aligned} e_m[n] - \hat{e}_m[n] &= (x_m[n] - \tilde{x}_m[n]) - (\hat{x}_m[n] - \tilde{x}_m[n]) \\ &= x_m[n] - \hat{x}_m[n] \; . \end{aligned}$$

This, in turn, implies that the optimal quantizer minimizing (6) also minimizes

$$\sum_{m=1}^{M} (x_m[n] - \hat{x}_m[n])^2$$

for fixed prediction coefficients $p_k[n]$.

We use the Lloyd algorithm [23] for the design of Q_n. This powerful technique is an iterative process which quickly converges at least to a local minimum. In order to ensure that it is not trapped in a poor local minimum, however, one must initialize it carefully. We adopt a splitting approach which starts with a 0-bit quantizer and gradually increases the number of bits used in the quantization. The advantage of the splitting approach is that it does not suffer from poor initialization, as the optimal 0-bit quantizer simply maps the whole real line to the *mean* of the distribution of $e_m[n]$ along m.

We also need to discuss here the structure of the quantizer Q_n to emphasize its ability to *reduce* the amount of data to be stored in the synopsis. The quantizer Q_n can be decomposed into an *encoder* f_n and a *decoder* g_n, i.e.,

$$Q_n(e) = g_n(f_n(e)) \; ,$$

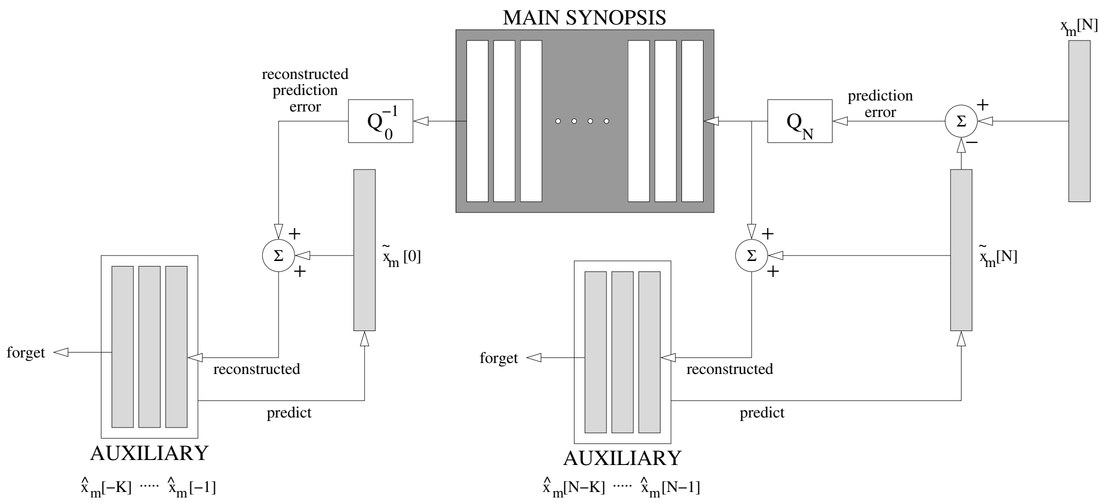


Fig. 3. Block diagram of the synopsis update algorithm.

where the output of f_n is the *bit description* of the prediction error e. Let

$$b_m[n] = f_n(e_m[n])$$

and

$$\hat{e}_m[n] = g_n(b_m[n]) = Q_n(e_m[n]) \ .$$

Note that at any time instant n, the same quantization functions f_n and g_n are used for all M streams, and therefore they require only negligible extra storage. The bit descriptions $b_m[n]$ for $1 \leq m \leq M$ and $0 \leq n \leq N-1$ constitute the main summary of the data.

The reconstruction of the data from the synopsis is performed as in (7) using $\hat{e}_m[n] = g_n(b_m[n])$. This process can be more explicitly stated as

$$\hat{x}_m[n] = \sum_{k=1}^{K} p_k[n]\hat{x}_m[n-k] + g_n(b_m[n]) \ . \tag{8}$$

Observe the recursive nature of (8), i.e., we need K previously reconstructed values of the data in order to compute $\hat{x}_m[n]$. This implies that the reconstructed values of the data at K time instants outside the sliding window, $\hat{x}_m[-1], \ldots, \hat{x}_m[-K]$, have to be stored as an auxiliary synopsis. As will be described in the next subsection, in order to update the synopsis efficiently at time $n = N$, we also need to store $\hat{x}_m[N-1], \ldots, \hat{x}_m[N-K]$ directly, even though those values can be computed from the rest of the synopsis via (8).

B. Constant-time Updating the Synopsis

Given the dynamic nature of data streams, having negligible update times is crucial for the summary to be useful for further analysis. This section outlines the update algorithm of PQ-stream which is constant in computational cost, and simple to implement. The block diagram of the update algorithm is shown in Figure 3. Below is a step-by-step verbal explanation of the actions taken by our algorithm:

1) Since the sliding window will be shifted one unit towards the future, the auxiliary synopsis $\hat{x}_m[-1], \ldots, \hat{x}_m[-K]$ needs to be updated as well. This is achieved by forgetting $\hat{x}_m[-K]$, shifting $\hat{x}_m[-1], \ldots, \hat{x}_m[-K+1]$ towards the back of the auxiliary synopsis window, and replacing $\hat{x}_m[-1]$ with $\hat{x}_m[0]$, which is to be computed using
$$\hat{x}_m[0] = \sum_{k=1}^{K} p_k[0]\hat{x}_m[-k] + g_0(b_m[0]) \ .$$
2) All stored parameters $p_k[n]$, f_n, g_n, and $b_m[n]$ for $1 \leq n \leq N-1$ are shifted once towards the past, thus forgetting old values of $p_k[0]$, f_0, g_0, and $b_m[0]$.
3) Similarly, the auxiliary sequences $\hat{x}_m[N-1], \ldots, \hat{x}_m[N-K+1]$ are shifted towards the past. The old values of $\hat{x}_m[N-K]$ are thus forgotten, and a new value for $\hat{x}_m[N-1]$ is to be computed in the following steps.
4) New coefficients $p_k[N-1]$ are to be computed for efficient prediction of newcoming samples y_m. This is accomplished solving (5) with $n = N-1$ and y_m replacing $x_m[N-1]$, i.e.,
$$\begin{bmatrix} p_1[N-1] \\ p_2[N-1] \\ \vdots \\ p_K[N-1] \end{bmatrix} = \mathbf{A}[N-1]^{-1} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_K \end{bmatrix} ,$$

where

$$d_k = \sum_{m=1}^{M} y_m \hat{x}_m[N-1-k] \; .$$

In fact, we do not need to invert $A[N]$. Since it is symmetric and approximately Toeplitz for large M, one can use the O(K^2) time Levinson-Durbin [11] recursion to solve (5).

5) The prediction error $e_m[N-1]$ is computed using (3) and (4), again replacing $x_m[N-1]$ with y_m.
6) A new quantizer Q_{N-1} is designed using $e_m[N-1]$ as training data and the values of f_{N-1}, g_{N-1}, and $b_m[N-1]$ are computed accordingly.
7) The reconstructed value $\hat{x}_m[N-1]$ is computed using (8).

Notice that the total time complexity of these steps is *independent* of N, the sliding window length. The "shifting" of the synopsis $\{p_k[n], f_n, g_n, b_m[n]\}$ can be implemented using pointers without physically shifting these parameters.

As described above, the update of the synopsis in the PQ-Stream technique is done in constant time in terms of the sliding window length N. Furthermore, it does not require the actual value of $x_m[0]$, the oldest sample in the window, for the updating procedure. In contrast, in order to update the stored coefficients with full precision in the DFT-based method, one needs to know $x_m[0]$, which must be forgotten after being processed in a truly one-pass algorithm. The solution proposed in [31] was to use "basic windows," which are obtained by further dividing the sliding window into windows of length b, and to store digests of these windows facilitating the update of the first few DFT coefficients of the whole sliding window. However, to amortize the storage cost, one must choose large enough b, which, in turn, will create delay problems. For the DWT method, it is even more difficult to update the wavelet coefficients, a simple shift in time could significantly change the coefficients. In practice, on the other hand, any $x_m[n]$ can be estimated using the transform coefficients themselves, and new coefficients can be approximated using this estimate and the new incoming sample. However, since it is the approximated coefficients that will be used for approximating the new $x_m[n]$, the error could eventually become large and a reinitialization might be needed.

C. Reconstruction and Utilization of the Synopsis

In previous sections, we explained our algorithms to generate and update the synopsis efficiently. We now describe how to reconstruct the actual data in the sliding window from the generated synopsis to execute online queries. Since $\hat{x}_m[N-1], \ldots, \hat{x}_m[N-K]$ are already stored, no further action is needed for the reconstruction of the required data. For the rest of the data elements in the sliding window, the reconstruction starts with the last forgotten entry and go back till the last entry or last queried entry. It is performed in a reverse order using (8), i.e.,

$$\hat{x}_m[n-K] = \frac{\hat{x}_m[n] - \hat{e}_m[n] - \sum_{k=1}^{K-1} p_k[n]\hat{x}_m[n-k]}{p_K[n]}$$

beginning with $n = N-1$ and until $n = K$. An important fact to remark here is that $p_K[n]$ must be non-zero. For numerical stability considerations in practice, an even more severe constraint is that $p_K[n]$ should not be very small. This can be ensured by not letting the prediction window length K be unnecessarily long. In fact, our experiments have shown that $K = 1$ is already very effective in practice, and that the optimal prediction coefficient $p_1[n]$ is usually close to 1.

Although queries involving other time intervals can also be answered, the recursive nature of the synopsis especially facilitates the queries involving most recent data entries. This extra support is, in fact, easily justified by the very reason why a sliding window is used: because recent values of the data are more significant. Hence, we will focus on queries involving the last L time instants $N-1, \ldots, N-L$, where $L \leq N$. Since the query involves only $x_m[N-1], \ldots, x_m[N-L]$ for $1 \leq m \leq M$, only those values need to be reconstructed, rather than the whole sliding window. If $L \leq K$, as discussed above, no further action is needed. If $L > K$, on the other hand, $L - K$ data elements needed to be reconstructed so, the reconstruction process will continue until $n = N - L + K$ instead of $n = K$. The process takes at most $O(KL)$ time.

As discussed in the introduction section, the application of such utilization is wide, such as clustering, indexing, query processing, etc. To evaluate the quality of the synopses, we will focus on query processing for clarity in the paper.

IV. Experimental Results

We performed experiments on real and synthetic data sets of multiple streams to demonstrate that PQ-Stream is generated and updated in real time and produces highly accurate summaries. We compared PQ-Stream with transform-based methods, DFT, DWT, 2-D DFT, and PCA in terms of recall and precision for a variety of queries. We also compared with SPIRIT [25], a recent technique to compute correlations in multiple streams by applying PCA and updating a set of principal components at each time instant. Since, the number of stream clusters can change as new data arrives, the number of components (hidden variables) is allowed to vary as needed. The weights of each stream on each hidden variable are stored and updated to reconstruct the actual values. However, the snapshot of the matrix needs to be stored for each corresponding time instant. This requires more space than the one required to save the original data.

In our experiments, the settings were favoring the methods in comparison. For example, since the focus of SPIRIT is monitoring, rather than summarization, we have not enforced any space limitations on it. Also, we assume, in favor of the transform-based methods (DFT, DWT, 2-D DFT, and PCA) that there exist very efficient algorithms that keep track of the top coefficients as time proceeds and old values of the streams are forgotten. Even when the DFT, DWT, 2-D DFT, and PCA coefficients are assumed to be efficiently updated in full precision and no space restriction is put on SPIRIT, our experiments show that PQ-Stream achieves in real-time a more accurate representation of the data.

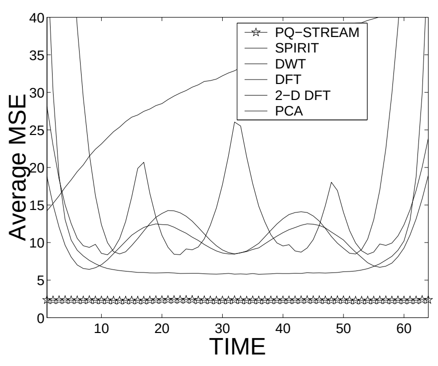
We first describe the data sets we used. We then show an illustrative example for effectiveness of PQ-Stream in signal representation quality. Then we show the results for generation and update of the synopsis, as well as the query processing over the generated synopsis.

A. Datasets

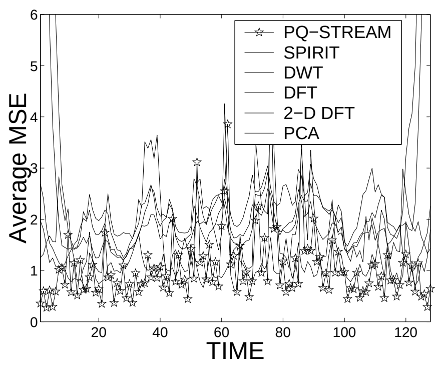
In our experiments, we used three datasets. The first dataset consists of daily stock values of 6,470 companies for 360 days. The second one is obtained from the National Climatic Data Center and consists of the temperature measurements for the year 2000 (366 days), sent from 5,009 weather stations worldwide. The last one is a synthetic data consisting of 100,000 streams of length 100, where the samples $x_m[n]$ are generated according to $x_m[n] = 0.99x_m[n-1] + w_m[n] + s_m[n]$, where $w_m[n]$ is a Gaussian noise independently drawn for each m and n with zero mean and variance 4, and $s_m[n]$ is a random spike with a 0.01 probability of occurrence. The function of $s_m[n]$ is to occasionally disrupt the predictability of the signal. The first data element of each stream is created randomly over a uniform distribution.

B. Effectiveness of PQ-Stream in Signal Representation Quality

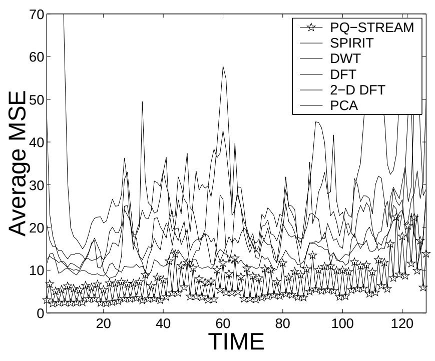
In this section, we demonstrate the superiority of the PQ-Stream scheme over current transform-based techniques in terms of average MSE. For comparison, we fix the total number of bits per stream to be used in the summarization of the data, thus using the same space complexity with transform based methods. We also take into account the utilized storage for the reconstructed version of the current time point, i.e. $\hat{x}_m[n]$. In Figure 4, we show the average MSE for PQ-Stream, DFT, DWT, 2-D DFT, PCA, and SPIRIT-based summarization techniques for all datasets.



(a) Synthetic Data



(b) Stock Data



(c) Weather Data

Fig. 4. Comparison of Average MSE by DFT, DWT, 2-D DFT, PCA, Spirit, and PQ-Stream methods with the same space complexity.

The prediction frame, K, is set to 1. In Figure 4(a), the sliding window length is $N = 64$, and a total bit quota of 96 bits/stream is used for summarization. 32 bits are used for the reconstructed version of the current time unit. We distribute the remaining 64 bits uniformly among 64 time instants by designing 1-bit quantizers for $e_m[n]$ at each time instant. An equal space complexity is achieved by storing top 3 Haar wavelet coefficients for the DWT, and PCA methods, and the first two frequency coefficients for the DFT method (since the second coefficient is complex). For 2-D DFT, we keep the greatest $3 \times 100,000$ coefficient values. SPIRIT maintains varying number of principal components (PCs) for each time point. The minimum number of PCs it can utilize is set to 3 for the synthetic data. The PQ-Stream technique clearly has less average MSE per stream compared to the DFT, DWT, 2-D DFT, PCA, and SPIRIT-based techniques. Figure 4(b) demonstrates a similar result on a real-life data set consisting of stock values. In this example, $N = 128$, and a total of 224 bits are reserved for each stream. Again 32 bits are utilized for the reconstructed version of the current time unit and the remaining 192 bits are distributed evenly among 128 time instants by alternating 1-bit and 2-bits for consecutive instants. With this space complexity, we can store 7 top Haar wavelet and PCA coefficients and first 4 DFT coefficients (three of them are complex numbers). For 2-D DFT, we keep the highest

$7 \times 6,470$ values. Similarly, the minimum number of PC, SPIRIT can maintain at each time instant is set to 7. For this dataset, PQ-Stream achieves the minimum average MSE for 80 time instants. The average MSE for the whole window is 1.04 for the PQ-Stream and it is 1.14 for the closest method, DWT. The setting for weather data is same as the one for the stock data, except the 2-D DFT technique. For this transformation, we keep the highest $7 \times 5,009$ values. As can be seen in Figure 4(c), PQ-Stream clearly minimizes the average MSE per stream the most for all time instants. These graphs not only show how superior the PQ-Stream is in terms of minimizing the average MSE per stream but also demonstrates the effect of number of bits in this error. In Figure 4(a), we are utilizing same amount of bit per time instant and the average MSE is not fluctuating compared to other methods. This is not the case for the other graphs. This can be seen clearly in Figure 4(c), for the instants using 1 bit the average error is greater compared to that of the ones using 2-bits. It is worth noting that SPIRIT utilizes at least N (i.e., 64 for synthetic data and 128 for stock and weather data) times more memory than other methods for all graphs.

C. The Synopsis and Its Update

The sliding window lengths and total bit quotas are fixed as $N = 128$ and $B = 224$ for the real data, and $N = 64$ and $B = 96$ for the synthetic one. The bit quotas are chosen so as to ensure that they can be fully utilized by transform-based methods to store 32-bit floating point coefficients. Since the first coefficient of DFT is always a real number, and the rest is in general complex, we reserved an odd multiple of 32 bits for summarization. In the PQ-Stream technique, we distribute $B - 32$ bits among N time instants evenly and the remaining 32 bits are utilized for saving the reconstructed version of the last time instant (since we use a prediction window of length $K = 1$ in our method). Recall that we have not enforced any space limitations on SPIRIT and it automatically decides the total number of principal components held for each instant. We only set the minimum number of PC it can maintain on behalf of this technique. Hence, we did not tabulate its parameter information in the results.

To verify our claims that the update of the synopsis is performed in a very time-efficient manner, we have physically measured the time spent on the update of the synopsis for the synthetic data set of size 100,000. A MATLAB code run on a computer with a 1.7GHz Pentium M processor updates the whole synopsis in 351.6ms. In other words, PQ-Stream is capable of handling about 284,000 stream updates per second. Therefore, even a non-optimized PQ-Stream code on a standard computer can handle the stock trading example mentioned in the introduction.

D. Query Models and Quality of Results

We used three types of queries involving last $L \leq N$ values of the streams, where L is picked randomly.

We call the first class of queries "momentum queries." An example for this type of query is: ***"Find all companies whose stocks appreciated value by 10% in the last two months."*** This query is particularly useful in testing how the resulting synopsis preserves the previously existing correlations between consecutive dimensions.

The second class is called "range queries." A sample query might be as follows: ***"Find all companies whose stock values remained in between 20 and 40 dollars per share in the last quarter."*** This type of query can be considered as an aggregation query: Find all companies whose stock values has max ≤ 40 and min ≥ 20 in the last quarter.

Finally, the third type of queries we used is commonly known as "similarity queries:" ***"Find the 25 companies whose stock value behavior were most similar to this waveform last week",*** or ***"Find the 25 companies whose stock value behavior were most similar to that of stock index YYZXX last week."*** The results for this query illustrate how much relative pairwise distances are preserved by the underlying summarization technique.

For each class of queries, we randomly generated 250 examples for the real datasets and 50 examples for the synthetic dataset, and measured the *recall* and the *precision* of the outcomes. Recall is the ratio of the number of relevant streams retrieved to all relevant streams, and precision is the ratio of number of relevant streams retrieved to number of streams retrieved. For the similarity search, the queries are randomly drawn over the data streams themselves and 25 nearest neighbors (25-NN) are asked.

Although the average recall and precision together indicate the quality of a particular summarization scheme, it is even more informative to have the standard deviations achieved for these metrics. The table below simultaneously depicts the average values and standard deviations in the format "avg | std". As can be seen from the table, the PQ-Stream method achieves high recall and precision values not only on the average, but also consistently for each query for the class of momentum queries.

Data Type	Accuracy	PCA	2-D DFT	DFT	DWT	SPIRIT	PQ-STREAM
Stock	Recall	.77 | .06	.28 | .09	.59 | .12	.86 | .07	.85 | .05	.87 | .03
Data	Precision	.85 | .03	.54 | .18	.59 | .21	.80 | .05	.82 | .05	.72 | .07
Weather	Recall	.15 | .10	.52 | .19	.48 | .26	.79 | .19	.78 | .19	.87 | .07
Data	Precision	.18 | .14	.84 | .14	.46 | .36	.80 | .18	.85 | .13	.83 | .10
Synthetic	Recall	.56 | .02	.31 | .10	.65 | .15	.71 | .06	.33 | .03	.88 | .01
Data	Precision	.76 | .03	.69 | .08	.57 | .09	.73 | .03	.70 | .03	.88 | .01

Similarly, for range queries, PQ-stream consistently outperforms the transform-based methods, as it achieves comparable recall (which is very high) and significantly higher precision. The reason why recall values are very high for the transform-based method is that they tend to smooth out positive and negative spikes in the signal, as they only keep low frequency coefficients.

Therefore, if a signal is in between the two values provided by the range query in the whole query window, it is very likely to stay there after being reconstructed from transform coefficients. But because of the same filtering property, the answer sets also include many false hits, thus resulting in poor precision. Average and standard deviation of the recall and precision values for the range query class are as provided below.

Data Type	Accuracy	PCA	2-D DFT	DFT	DWT	SPIRIT	PQ-STREAM
Stock Data	Recall	.98 | .01	.94 | .04	.97 | .03	.98 | .01	.97 | .02	.97 | .01
	Precision	.92 | .05	.91 | .04	.87 | .05	.88 | .05	.94 | .03	.97 | .02
Weather Data	Recall	1.0 | .01	.97 | .02	1.0 | .01	1.0 | .01	1.0 | .01	.95 | .01
	Precision	.81 | .05	.90 | .02	.72 | .07	.71 | .06	.84 | .04	.97 | .01
Synthetic Data	Recall	.97 | .01	.88 | .10	.98 | .02	.99 | .03	.81 | .12	.94 | .01
	Precision	.70 | .01	.75 | .08	.53 | .05	.56 | .07	.80 | .03	.95 | .01

Recall that number of streams retrieved for 25-NN queries is 25 and number of relevant streams is also 25, hence, the recall and precision is equal to each other for this query type. For 25-NN queries, the average and the standard deviation of the recall and precision values are provided below.

Data Type	Accuracy	. PCA	2-D DFT	DFT	DWT	SPIRIT	PQ-STREAM
Stock Data	Recall = Precision	.75 | .16	.43 | .25	.64 | .23	.75 | .17	.71 | .16	.81 | .13
Weather Data		.61 | .21	.61 | .18	.34 | .29	.55 | .25	.63 | .18	.80 | .18
Synthetic Data		.09 | .14	.20 | .16	.07 | .12	.14 | .16	.01 | .04	.55 | .16

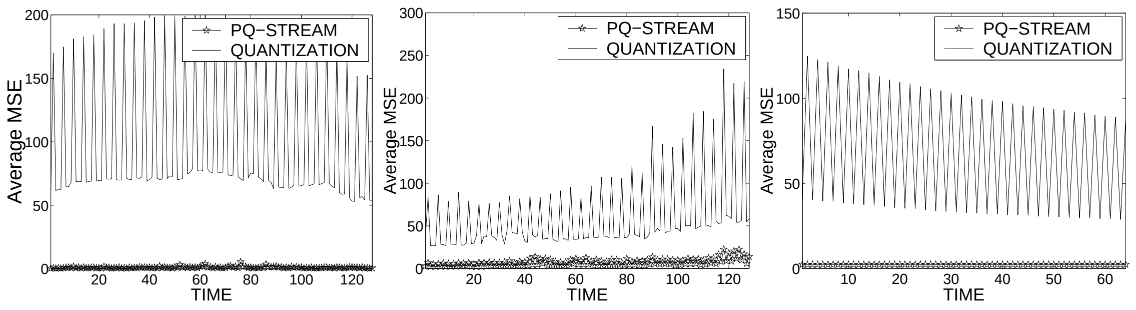
E. Analysis of PQ-Stream Components

We analyzed the effect of the two main components of PQ-Stream, prediction and quantization, when used separately. In the first set of experiments, we applied prediction followed by 0-bit quantization using 0 as the only reconstruction level. In the second set, we quantized each data element by using the default setting (i.e., 224 bits for real datasets and 96 bits for the artificial dataset) as in the VA-file method [30]. Although prediction and quantization individually achieves reasonable recall and precision values for certain cases in all types of queries, PQ-Stream consistently and significantly outperforms both. In many cases, PQ-Stream is able to capture query answers which are missed by both of them individually. Similarly, false hits introduced by each of the techniques are removed by PQ-Stream. The highest individual contributions of each of prediction and quantization were in range queries, which are summarized below.

Data Type	Accuracy	Prediction	Quantization	PQ-STREAM
Stock Data	Recall	.5137	.4650	.9741
	Precision	.5041	.5647	.9685
Weather Data	Recall	.9008	.6151	.9448
	Precision	.7009	.8189	.9664
Synthetic Data	Recall	.2144	.0277	.936
	Precision	.5163	.0399	.9518

For momentum and similarity queries, the improvement of PQ-Stream over each component are even more significant. For range queries, prediction usually achieves more accurate results. This is because the range of the predicted values are usually close to the range of the original values. For momentum and similarity queries, influence of quantization is more than that of prediction. Since similar data are quantized to similar values, such comparison based queries perform better under quantization.

To analyze the effect of prediction on the quantization, we compared the average MSE achieved by quantization with and without prediction. As can be seen in all subgraphs of Figure 5, prediction followed by quantization achieves much fewer average MSE than the plain quantization. The main premise of PQ-Stream technique is not that all streams rise and fall in lockstep. The main assumption of it is that there is high correlation between nearby data elements of each stream, and hence the difference between the consecutive elements has a narrower range than the original elements. Prediction narrows down the range of this difference and error of the prediction is given as the input to the quantizer. The quantizer is built in an MSE minimizing way. Hence, small number of bits, i.e. $3/2$ bits/time instant, is capable of capturing the changes for each time unit.



(a) Stock Data (b) Weather Data (c) Synthetic Data

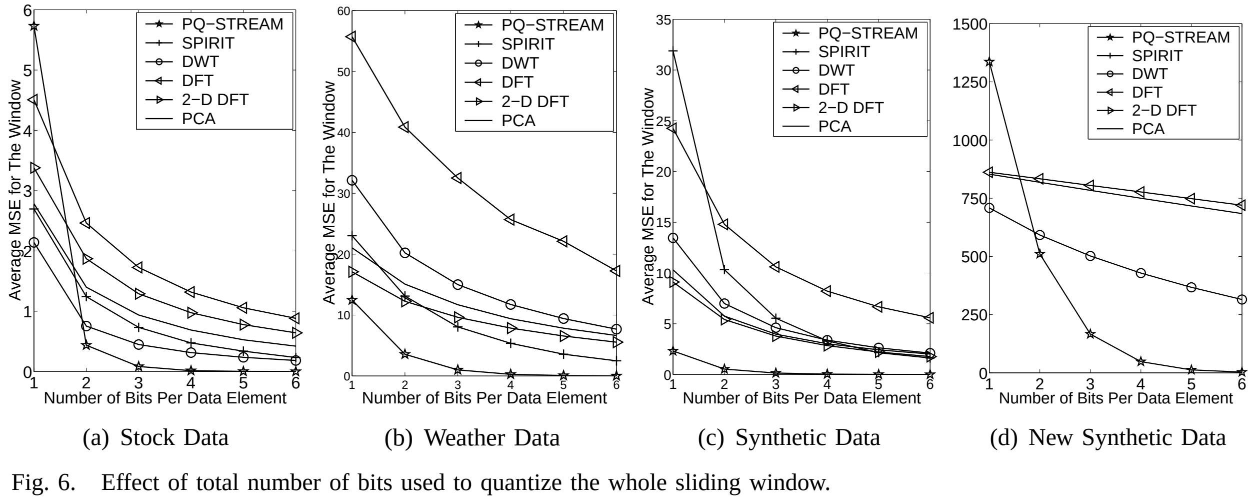
Fig. 5. Effect of Prediction on Quantization.

F. Space-Accuracy Trade-off

It is clear that the performance of PQ-Stream varies depending on the number of bits used for quantizing the sliding window: more bits translate into more precision of the quantized error and therefore into more accuracy when reconstructing the stream. To measure the effect of the number of bits on the average MSE over the sliding window, we designed 6 configurations for each data set. For the configuration s, we are using s bit(s) per time instant, where $s \in [1, 6]$. The total number of bits utilized for configuration s is $N \times s + 32$, i.e. for the artificial data (N=64) it is $64s + 32$ bits and for the real datasets, where N is 128, it is $128s + 32$ bits. For DWT and PCA we are storing the top $2s+1$ coefficients for the synthetic dataset and $4s+1$ top coefficients for the real ones. For SPIRIT these are the minimum number of PCs that can be stored. The top s+1, and $2s + 1$ coefficients are stored for synthetic data and real datasets respectively for DFT. We are keeping the top $(2s + 1) \times 100,000$ coefficient values for the synthetic data and $(4s + 1) \times M$ top values for real datasets, where M is the number of streams in the dataset for 2-D DFT.

To study the performance of PQ-Stream on datasets not having correlation between nearby points, we generate another artificial dataset. The configurations, the number of streams and the number of time instants for this new dataset, is same as the ones for the old one. The samples $x_m[n]$ are generated according to $x_m[n] = min_m + |std_m| \times |w_m[n]|$, where min_m is created randomly over a uniform distribution over $[0, 100]$, std_m is independently drawn for each m from Gaussian distribution with zero mean and variance 2500, and $w_m[n]$ is from Gaussian distribution with 0 mean and variance 1.

The performance of all techniques on average MSE for these 4 datasets are depicted in Figure 6. As can be seen in all subgraphs, increasing the number of bits per time instant is leading to decrease the average MSE for each technique. After a certain number of bits, this average error approaches to zero, i.e. after 4 bits for the stock and the weather datasets, for our technique, PQ-Stream. This is not the case for the transform based techniques even for saving 25 coefficients, out of 128, for the real datasets and 13 coefficients, out of 64, for the synthetic ones. In Figure 6 (d), results for 2-D DFT and SPIRIT are not shown because the smallest error they achieve is greater than 1500 and adding them in the graph is leading less detail on the comparison between other methods.



(a) Stock Data (b) Weather Data (c) Synthetic Data (d) New Synthetic Data

Fig. 6. Effect of total number of bits used to quantize the whole sliding window.

G. PQ-Stream over Clustered Streaming Data

Data stream sources can be grouped into clusters to further improve the quality of the prediction model, the summaries and the accuracy of queries. For example, sensor nodes can be naturally grouped with respect to their spatial locations. We analyze the effect of building a PQ-Stream for each stream cluster. We applied K-Means [23] clustering algorithm over the real data sets for 1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 clusters. The clustering was done after a training phase. Up to a certain time point, a single PQ-Stream was maintained for all streams. As only the data under the sliding window can be reconstructed from the main synopsis, this point was selected as the the first time the sliding window is full, i.e. time $n = 128$. Then, the data elements under the sliding window were reconstructed for all streams and clustering was done on top of this data. The experiments were performed using the same number of bits, 224, as used for the original setting.

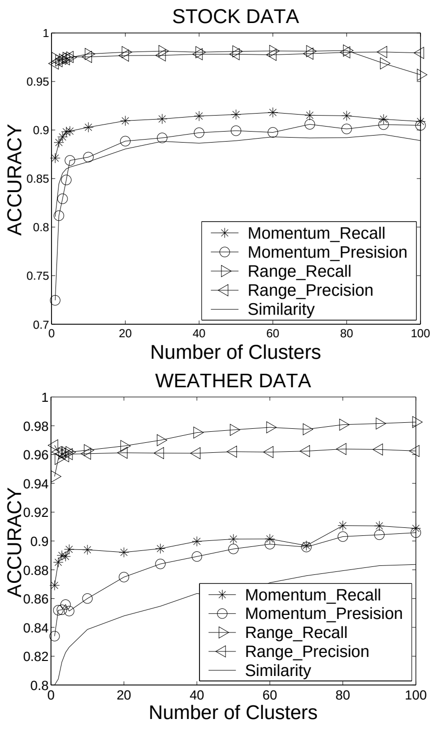


Fig. 7. Effect of establishing a PQ-Stream for each stream cluster.

As shown in Figure 7, the accuracy mostly increases with increasing the number of clusters. The most significant improvement is observed for similarity queries. There has been a sharp increase for up to 5 clusters in stock and 10 clusters in weather data. This improvement can also be translated into a reduction in space with comparable accuracy. Specifically, after the clusters were formed, we decreased the total number of bits from 224 to 160 and obtained the results summarized in the following table.

Query Type	Accuracy	Stock Original 224 bits	Stock Clustered 160 bits	Weather Original 224 bits	Weather Clustered 160 bits
Momentum	Recall	.8711	.8715	.8692	.8750
	Precision	.7246	.8111	.8339	.8316
Range	Recall	.9741	.9619	.9448	.9526
	Precision	.9685	.9620	.9664	.9511
Similarity	Recall=Pre-cision	.8107	.8221	.8000	.7618
		.8107	.8221	.8000	.7618

V. Related Work

V-Optimal histogram was introduced by Ioannidis and Poosala in [17]. The problem is to find the best piecewise constant representation of the data with at most B pieces, in a way that minimizes the sum of squares error between the data and the representation for given n numbers and a constant B. Recently, Guha et al. [13] provided an O(n^2B) time O(n) space algorithm to find the optimum B bucket synopsis. For the sliding window model, the first data stream histogram algorithm was presented by Guha and Koudas [15]. For each new data point the algorithm uses O($N(B^3/\epsilon^2)log^3(N)$) time to find the ϵ-approximate histogram. It requires O(N) storage space since it saves a structure for each data point in the window. For each newly arrived data element, the histogram is recomputed for the last window. Thus, those data elements are needed to be saved and retouched for each new element. This leads to a multiple-pass update algorithm. Our approach, PQ-Stream, utilizes O(N) bits, but retouches each reconstructed element K times.

If the data is arriving faster than it can be processed, a small random sample S of the data is often used to execute queries [2]. Distribution of the data in the sliding window can be different than that of the whole stream. Hence, a sample from the last N dimensions should be kept. In [3], the authors studied the problem of maintaining a uniform random sample of items within the current window. The provided results do not give any bound for estimating similarity/distance.

Sketch of an n dimensional vector is a k dimensional vector computed by k dot products. To compute each dimension of a sketch an n dimensional random unit vector is utilized. A bound on Euclidean distance, depending on k, was provided by Johnson-Lindenstrauss Lemma [18]. Indyk et al. [16] proposed that with probability of 1/2, success on error bound can be provided depending not only on k but also on the size of the vector set. However, for $M >> N$,the required k for the success of $1/2$ is close to N. To extract representative patterns from a single massive time series data, Indyk et al. [16] employed convolution to compute all possible sketches of the same size. As the possible number of subsequences and the required size for each sketch are both N; the required space, O(N^2), is greater than the size of the actual data. Due to the size of each sketch, the comparison between two sketches for any arbitrary L will take O(N) time. In contrast, PQ utilizes O(L) time for the same operation. Hence, keeping such a synopsis would be infeasible.

Datar et al. [8] provided efficient algorithms to obtain L$_p$ norms (for $p \in [1, 2]$) of the vector under the sliding window. Their model is time, and space efficient for the queries on the statistics of the last N elements of a single vector such as min/max, sum, average and number of distinct values. However, if R, range of the data elements, is poly(N) then all N elements have to be stored [8] for Min/Max queries. This approach is not feasible for the queries considering similarities between multiple streams, i.e., similarity queries. The algorithm combines the buckets and saves the statistics of the combined bucket. Therefore, it is not viable for comparing two different elements of the same stream, momentum queries. On the other hand, PQ-Stream can be used for any purpose that requires the actual data.

Our approach provides a flexible synopsis that can be used to execute queries over last L data elements, where $L \in [1, N]$. It can handle an arbitrary L since it is based on reconstructing the actual data from the synopsis at hand. In contrast, most of current approaches require generation and maintenance of a synopsis for each possible L. An exception to this is transform-based methods, which are used for comparison in experimental section.

VI. Conclusion & Discussion

We proposed PQ-Stream, which is based on predictive quantization, as a novel methodology for online summarization of multiple data streams. We focused on the model where the synopsis of multiple data streams over a sliding window of most recent entries is to be computed in a one-pass fashion. In this streaming scenario, predictive quantization is shown to be more advantageous over transform-based techniques, as the amortized time complexity of the synopsis update is $O(1)$ for K =1, i.e., independent of the sliding window length. Moreover, the predictive quantization achieves the same coding gain (i.e., representation quality) as any optimal transform-based method when the prediction window length is large enough. Complementing this theoretical result, we have shown that even when used with short prediction windows (as short as $K = 1$), PQ-Stream is superior to current transform-based techniques both in data representation quality and in precision and recall for various types of queries. We have provided step-by-step algorithms for generating and updating the dynamic summary over data streams, and for processing of queries. With PQ-Stream the queries are answered in $O(L)$ time where L stands for the length of the query window, which can be over any subsequence of multiple streams. Although we focused on the multiple data stream model, PQ-Stream is also effective for traditional high dimensional databases. Even when the current approaches use extra preprocessing over the data, PQ-Stream achieves provably better results with its online algorithm.

There are several interesting directions for use of PQ-Stream in other emerging data management applications. An immediate use is *distributed compression*, where each stream is compressed on-the-fly as it is generated by the data sources, e.g., sensor nodes. The main goal of such a process is to reduce the amount of data transferred between nodes, hence the energy consumed to send them. Sensor nodes are already grouped into flat or hierarchical structures, mostly with respect to their spatial locations. Building a PQ-Stream for each stream cluster improves the quality of the prediction model, the summaries and the accuracy for each query types. After collecting the new data elements from its members, each group leader can compute the prediction coefficient, summarize the collected data using PQ-stream, and transmit to a central processing unit only this summary. Besides its use in data transmission, the already computed prediction coefficient can also be effectively used for *real-time change detection*. A change on the prediction coefficient implies an overall change on the streams from the corresponding group of data sources, such as spatially close sensors. It is highly efficient and simple to define and monitor changes on a single parameter. The same process is applicable to sensors measuring multiple attributes. In this case a prediction coefficient will be used for each attribute, resulting a vector for all attributes. The parameters of the prediction function can be used for other data mining purposes as well. For example, having similar coefficients for different regions may be an indication of similarity in their environmental characteristics.

REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *4th Int. Conference on Foundations of Data Organization and Algorithms*, pages 69–84, 1993.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16, New York, NY, USA, 2002. ACM Press.

[3] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *ACM-SIAM Symposium on Discrete Algorithms(SODA)*, 2002.

[4] S. Babu and J. Widom. Continuous queries over data streams. *SIGMOD Record*, 30(3):109–120, September 2001.

[5] S. Berchtold, C. Bohm, H. Jagadish, H. Kriegel, and J. Sander. Independent quantization: An index compression technique for high-dimensional data spaces. In *Proc. 16th Int. Conf. on Data Engineering*, pages 577–588, San Diego, CA, 2000.

[6] A. Bulut and A. Singh. Stardust: Fast stream indexing using incremental wavelet approximations. Technical Report TRCS03-24, Dept. of Computer Science, Univ. of California, Santa Barbara, 2003.

[7] E. Cohen and M. Strauss. Maintaining time-decaying stream aggregates. In *Proc. ACM Symp. on Principles of Database Systems*, pages 223–233, San Diego, CA, 2003.

[8] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *ACM-SIAM Symposium on Discrete Algorithms(SODA)*, 2002.

[9] A. Dobra, M. Garofalakis, J. E. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *ACM Sigmod International Conference on Management of Data*, pages 61–72, Madison, Wisconsin, June 2002.

[10] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. El Abbadi. Vector approximation based indexing for non-uniform high dimensional data sets. In *Proceedings of the 9th ACM Int. Conf. on Information and Knowledge Management*, pages 202–209, McLean, Virginia, November 2000.

[11] A. Gersho. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, MA, 1992.

[12] A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Straus. Surfing wavelets on streams: One pass summaries for approximate aggregate queries. In *VLDB*, 2001.

[13] S. Guha. Space efficiency in synopsis construction algorithms. In *Very Large Databases(VLDB)*, pages 409–420, 2005.

[14] S. Guha, D. Gunopulos, and N. Koudas. Correlating synchronous and asynchronous data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 529–534, Washington, D.C., 2003.

[15] S. Guha and N. Koudas. Approximating a data stream for querying and estimation: Algorithms and performance evaluation. In *IEEE International Conference on Management of Data*, San Jose, California, June 2002.

[16] P. Indyk, N. Koudas, and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In *In Proc. of the 26th Int. Conf. on Very Large Data Bases(VLDB)*, September, 2000.

[17] Y. E. Ioannidis and V. Poosala. Balancing histogram optimality and practicality for query result size estimation. In *SIGMOD ’95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 233–244, New York, NY, USA, 1995. ACM Press.

[18] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982), Amer. Math. Soc., Providence, R.I.*, pages 189–206, 1984.

[19] J. Kang, J. F. Naughton, and S. D. Viglas. Evaluating window joins over unbounded streams. In *ICDE*, pages 560–571, Bangalore, India, March 2003.

[20] F. Korn, S. Muthukrishnan, and D. Srivastava. Reverse nearest neighbor aggregates over data streams. In *Proceedings of the Int. Conf. on Very Large Data Bases*, Hong Kong, China, August 2002.

[21] B. Krishnamurthy, S. Sen, Y. Zhang, and Y Chen. Sketch-based change detection: Methods, evaluation, and applications. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, 2003.

[22] X. Liu and H. Ferhatosmanoglu. Efficient k-nn search on streaming data series. In *International Symposium on Spatial and Temporal Databases*, pages 83–101, Santorini, Greece, July 2003.

[23] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:127–135, March 1982.

[24] S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive hands-off stream mining. In *VLDB*, pages 560–571, Berlin, Germany, September 2003.

[25] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 697–708. VLDB Endowment, 2005.

[26] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima. The a-tree: An index structure for high-dimensional spaces using relative approximation. In *Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000*, pages 516–526, Cairo, Egypt, 2000.

[27] N. Thaper, S. Guha, P. Indyk, and N. Koudas. Dynamic multidimensional histograms. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 428–439, Madison, WI, June 2002.

[28] P. A. Tucker, D. Maier, T. Sheard, and L. Fegaras. Exploiting punctuation semantics in continuous data streams. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):556–568, 2003.

[29] S. Viglas and J. F. Naughton. Rate-based query optimization for streaming information sources. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Madison, Wisconsin, June 2002.

[30] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the Int. Conf. on Very Large Data Bases*, pages 194–205, New York City, New York, August 1998.

[31] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, Hong Kong, China, August 2002.