A Fast Approximation Strategy for Summarizing a Set of Streaming Time Series

Alice Marascu
Inria Sophia Antipolis
AxIS Project-Team
2004, route des Lucioles
06902 Sophia Antipolis
alice.marascu@inria.fr

Florent Masseglia
Inria Sophia Antipolis
AxIS Project-Team
2004, route des Lucioles
06902 Sophia Antipolis
florent.masseglia@inria.fr

Yves Lechevallier
Inria Paris - Rocquencourt
AxIS Project-Team
11 domaine de Voluceau
78153 Le Chesnay
yves.lechevallier@inria.fr

ABSTRACT

Summarizing a set of streaming time series is an important issue that reliably allows information to be monitored and stored in domains such as finance [12], networks [2, 1], etc. To date, most of existing algorithms have focused on this problem by summarizing the time series separately [12, 4]. Moreover, the same amount of memory has been allocated to each time series. Yet, memory management is an important subject in the data stream field, but a framework allocating equal amount of memory to each sequence is not appropriate. We introduce an effective and efficient method which succeeds to respond to both challenges: (1) a memory optimized framework along with (2) a fast novel sequence merging method. Experiments with real data show that this method is effective and efficient.

1. INTRODUCTION

Recently, summaries of time series have drawn high attention because of their multiple applications and computation complexity. Several important applications related to this topic are querying [8], mining [4, 9, 7] and forecasting data [5, 2]. The computation difficulty is related to the very large volume of data that are continually produced at a very high rate. Actually, new data arrive as a stream, so the amount of computation time per data must be low. Furthermore, as processed elements are discarded or archived and the size of a data stream is unbounded this will lead to a memory problem. We introduce a memory optimized framework along with a new representation of time series that allows fast computing thanks to straight formulas. Let us suppose we have an available amount of memory M. We want to trace in time the behavior of maximum n time series; then we can imagine the available memory as a matrix with n rows. If the memory is uniformly distributed, we can keep $t = M/n$ values per time series; more precisely, each row would keep maximum t values. After t entries, the available memory M is full, so we have to apply an approximation algorithm.

There exist several techniques trying to answer to this problem.One class of techniques uses a local error optimization. More precisely, when for a sequence the available memory (space) is exceeded, a compression of this sequence must be done. Another class of techniques takes into account the passing time and uses a decaying factor. In this case, the older the value, the less important it is. Therefore, the older values should be merged more often than the recent ones (usually a logarithmic scale is used). In fact, most of existing solutions to the problems raised by the data stream history management belong to the second class [3, 6, 11]. The main idea of these methods is that, according to the human memory, we are generally more interested in recent events than we are in older ones. As a matter of fact, the human memory keeps a detailed report of the "salient" events but a careless record of the "unimportant" events (i.e. an accident versus a breakfast). Based on this principle, we propose a novel approach where the most important events are kept with high fidelity, while the less important events are kept at a coarser resolution. For this purpose we present GEAR (Global-Error Aware Representation), a fast on-line approximation algorithm. *The main feature of GEAR is its capacity to find the best approximation possible related to the global framework; moreover, it is in real time.* Since data streams require to process the data as fast as possible, *we also propose a new representation* which gives similar results (compared to traditional PLA) but involving *fewer variables and operators*. MITH (MIddle THrough), our novel representation, goes across the segments middles, involving a novel formula to calculate the line's coefficients.

The rest of this paper is organized as follows. Section 2 gives the definitions of our problems and Section 3 gives an overview of related works. In Section 4, we present our framework. Section 5 gives our new and fast approximation technique and Section 6 provides a set of experiments designed to evaluate the effectiveness and efficiency of GEAR and MITH.

2. PROBLEM DEFINITION

In this section we introduce the concept of Global Error-Aware Representation and define our research challenges.

2.1 Global Error-Aware Approximation

Let $T[1..n]$ be a set of n time series that we want to monitor. Let $T[j]$ be the j^{th} time series in T. Then, at a step s, $T[j] = [j_1, ..., j_s]$ is the set of observations (measures) for

SAC'10 March 22-26, 2010, Sierre, Switzerland.

$T[j]$. Let us consider that we are provided with an amount of memory of size M (that means we can keep in memory M segments or blocks or values). We consider the case of discrete time. At each step we have a value for each time series (if the time series doesn't appear, we keep a zero value, otherwhise, we keep its value). The maximum value of s to store all the time series without any compression is $s = M/n$. When $(s \times n) > M$, the representation of the original data has to be compressed and when $(s \times n) \gg M$ the compression cannot be without loss.

A popular representation of time series is Piecewise Linear Approximation (PLA). Let $A[1..n]$ be the set of approximations calculated on n time series. Let $S[i]$ be the size of $A[i]$ (*i.e.* the number of memory blocks used by $A[i]$, where a memory block represents a segment). We know that $\sum_{i=1}^{n} S[i] = M$. Let $E[i]$ be the error of $A[i]$ with respect to the original data. Let $E(A)$ be the global error of $A[1..n]$. Then, $E(A) = \sum_{i=1}^{n} E[i]$. Previous research considered either 1) how to optimize the error of a single PLA of a single time series (*i.e.* optimize $E[i]$ by merging segments of $A[i]$). or 2) how to give greater importance to recent events.

The obtained representation is such that $\forall i, j \in [1..n], S[i] = S[j]$ (i.e. the same amount of memory is used for all series). However, to the best of our knowledge, there has been no previous work on such an optimization for a set of streaming series with a global-error aware approach (*i.e.* optimizing $E(A)$ by merging and re-allocating segments in A at each step s). *The first problem discussed in this paper is "how to find the optimal distribution of M memory blocks in order to have the least global error $E(A)$ in an ongoing process.*" Solving this problem calls for a merging strategy.

2.2 Merging Segments

As explained in [8, 4], the PLA of two segments can be computed in O(1). Starting from the PLA, we propose a fast formula involving only a few parameters and operators. *Furthermore, we reduce the number of computational operations for the new approximation of two segments.* Our proposal of a new representation will be explained in Section 5.

3. RELATED WORKS

In [7], the authors present a symbolic representation of time series and the associated algorithm (SAX). SAX offers a discretization of the original data into symbolic strings, allowing manipulations on a symbolic representation, rather than on the original data. [7] introduces a function that returns the minimum distance between the original time series of two symbolic representations. In [10], the authors propose *segmented-means*, a method intended to extract the feature vector of every sequence of a time series set. For each time sequence, *segmented-means* first partitions it into equal length segments. Then, for each segment, the extracted feature is the *mean* of this segment. An important advantage of [10] is the suitability of their method for any $\mathcal{L}_p$ norm.

Techniques for incremental regression on time series are given in [4] and [8]. In [4], the authors propose an interesting principle to merge adjacent segments in linear time. Since their objective is to obtain such summaries depending on the user's constraints, they do not focus on the optimization of any error measure. In [8], the authors borrow the segment merging principle from [4] and provide an interesting theorem showing that *"the error of the line segment approximating all the original data points can be computed as the sum of the errors of the two individual line segments and the error between those two line segments and the line calculated based on those two."* Then, [8] develops studies on how to consider a user's decaying function in the regression.

Managing a time series from its first value to the current one is a feature of landmark models [4]. In [4], the authors propose to store the summaries of streaming time series in a cube. They use the decaying model of Tilted Time Windows (TTW) in order to reduce the cost of computing and materializing the values at a multi-dimensional space. To conclude this section on related works, let us cite [12], which deals with multiple streaming time series. The authors propose the StatStream system to monitor the behavior of a set of streaming time series and to detect stream pairs having a correlation coefficient larger than a user specified threshold. The comparison is made on reduced representations (DFT) of the raw data streams. The correlation are extracted thanks to pair-wise statistics that are computed in a incremental fashion (in constant time) for any pair of streams.

4. GEAR: MOTIVATION AND PRINCIPLE

In subsection 4.1 we present the general principle of our approach to summarize a set of streaming series. We also discuss the advantages of such a strategy compared to existing ones in subsection 4.2.

4.1 General Principle

Let us consider $T[1..n]$, the set of time series given in the problem definition. At each step s (a new value is added to each time series: zero, if the time series does not appear,otherwhise, its value), we want to update the representations and their error rates in our data structure. In order to maintain a globally satisfying error rate, we need to choose the representations that minimize this error. The main idea is that a representation with a low error rate will have higher tolerance to merging two of its segments (compared to a representation with a high error rate). When $(s \times n) > M$ we need to release blocks (segments) in order to take into account the new values. Therefore, at step s, for each new value:

1. Let r be the representation having the minimum local error. Then r will be compressed (two of its segments are merged);
2. $E(r)$, the local error of r, is updated;
3. $E(A)$, the global error, is updated.

4.2 Motivation and Generic Algorithm

Approximations based on a decaying factor have some advantages. One of them is that they give a good precision for recent events. However, they also have some drawbacks, such as being blind to important stream fluctuations. Fraud detection, for instance, might need to access previous anomalies in the history. An anomaly is a salient event and for security purposes it should not be merged with any other data, even when it becomes old. Our goal is to give more importance to salient events, whatever their age. We believe that querying the history of a streaming time series set must benefit from a model that carefully distributes the precision on periods that require it. Figure 1 gives the generic algorithm of GEAR. Without loss of generality, we

Algorithm GEAR
In: DS, a data stream of n time series and M, the number of segments allocated to the model.
Out: at any step s, $R[1..n]$, is the set of segments representing DS with an optimal global error. $R[i].last$ and $R[i].penultimate$ are the last and penultimate segments of $R[i]$.

1. While $(s \times n) < M$, assign one segment to R for each new value of DS
2. Build C, the heap storing the costs of merging segments in R. // $C[1]$ *is the lowest value in* C.
3. When a new set of n values arrives (at each step s), $\forall i \in [1..n]$
   (a) $Buffer$ = a segment assigned to $DS[i][s]$
   (b) $ErrorNew = ErrorEval(R[i].last, Buffer)$
   (c) If $(ErrorNew < C[1])$
      i. Merge the last segment of $R[i]$ with $Buffer$
      ii. Update the error of $R[i]$ with $ErrorNew$
   (d) Else, let t and t' be the couple of segments in $C[1]$
      i. Merge t and t' into t'' and update the error of the corresponding representation
      ii. Update C with the costs of merging t'' with the previous and next segments
      iii. Add $Buffer$ to $R[i]$
4. Update C with the cost of merging $R[i].last$ with $R[i].penultimate$
5. remove $C[1]$

Figure 1: The generic algorithm of GEAR.

consider that we use a heap (*i.e.* C) to sort the couples of segments by increasing order of merging errors. The space complexity of GEAR is O(M). Actually, in order to obtain the result described in 4.1, we need to evaluate for a couple of segments in a given representation the new error that would result from merging these segments. *In a streaming environment each optimization of this evaluation should be much appreciated.*

We compare PLA and MITH (our new representation, based on the middles of the considered segment), when used to evaluate or merge a couple of segments. [8, 4] give formulas for calculating the PLA of two segments in O(1) and for calculating the new error in O(1). Section 5 will give the details of our approach for calculating the MITH representation of two segments and how to evaluate the corresponding error. Therefore, each of the representations presented in [8, 4] and 5 might be used when merging two segments and updating the error of the corresponding time series in GEAR.

5. MITH: A NEW REPRESENTATION OF TIME SERIES

Even if the merging operation is linear in time, a difference in execution times due to less operations might be crucial for data streams. In this section we propose a novel representation with a innovative idea based on the following observation: when the cost of merging two segments is low, the PLA line crosses the segments near their middles.

Despite its natural and instinctive characteristics, no previous works have exploited this principle.

We consider the case of the linear regression model $Y = \alpha.X + \beta$, where α is the slope and β is the intercept. In order to estimate α and β, we have $((i, j_i), i = 1, \ldots, s)$ a time series on set $S = \{i | i = 1, \ldots, s\}$. Let $\bar{j}$ be the means of the sample $(j_1, \ldots, j_s)$, then $\bar{j} = \sum_{i=1}^{s} j_i/n$. The idea of least squares estimation is to minimize the sum of squared errors: $E(S) = \sum_{i=1}^{s} \varepsilon_i^2 = \sum_{i=1}^{s} (j_i - \alpha.i - \beta)^2$

A linear fit for a time series j_i for $i \in [1, s]$ is the linear regression equation $Y = \alpha.X + \beta$. The parameters α and β are chosen to minimize the residual sum of squares $E(S)$. These parameters can be obtained as follows: $\alpha = \frac{\sum_{i=1}^{s}(i-\bar{i})(j_i-\bar{j})}{\sum_{i=1}^{s}(i-\bar{i})^2}$ and $\beta = \bar{j} - \alpha.\bar{i}$ where $\sum_{i=1}^{s}(i-\bar{i})^2 = (s-1)s(s+1)/12$ and $\bar{i} = (s+1)/2$.

5.1 Merging Two Segments

In the traditional PLA of a time series, each segment lowers the sum of squared errors between the representation and the original values. Our proposal is to give a rather intuitive, though effective, approximation. MITH, our new representation, is the line which cuts the original two segments through their middles. This representation has two main advantages:

1. It does not require to calculate numerous variables, like the ones expressed in [8, 4], thus allowing a fast computing;
2. It has a natural interpretation. Even if it does not lower the sum of squared errors, it is easy to see that our representation gives a very reliable approximation of the original data. Actually, sum of squared error values are used when one wants to set a high penalty for being further away from outliers, which is not always the desired result when representing the original values of streaming time series. As we will show, representations obtained by MITH and PLA are comparable. However, we want our proposal to be less sensitive to outliers. Therefore, regarding the error, the representation of MITH is a good compromise between $\mathcal{L}_1$ and $\mathcal{L}_2$ norms. Actually, our experiments will show that MITH has better results when the error is measured in $\mathcal{L}_1$ norm.

Let $(\bar{i}_1, \bar{j}_1)$ and $(\bar{i}_2, \bar{j}_2)$ be the middle points of S_1 and S_2 with $\bar{i}_1 = (k+1)/2$ and $\bar{i}_2 = (s+k+1)/2$. The equation of segment S is obtained by solving the following set of linear equations: $\bar{j}_1 = \alpha.\bar{i}_1 + \beta$ and $\bar{j}_2 = \alpha.\bar{i}_2 + \beta$ where $\alpha = \frac{\bar{j}_1 - \bar{j}_2}{\bar{i}_1 - \bar{i}_2}$, $\beta = \frac{\bar{j}_2.\bar{i}_1 - \bar{j}_1.\bar{i}_2}{\bar{i}_1 - \bar{i}_2}$ and the new middle point $(\bar{i}, \bar{j})$ of S is given by $\bar{i} = (s+1)/2$ and $\bar{j} = \alpha.\bar{i} + \beta$. It is straightforward to demonstrate that $\bar{j}$ can be computed by $\bar{j} = \frac{k.\bar{j}_1 + (s-k)\bar{j}_2}{s}$. The previous formula proves that the middle point of S is the mean point of S and verifies: $\bar{i} = \frac{1}{s}.\sum_{i=1}^{s} i$ and $\bar{j} = \frac{1}{s}.\sum_{i=1}^{s} j_i$ An interesting property of that point is that it belongs to the linear regression line. The relation between the slopes of PLA and MITH lines can be measured by:

$$\begin{aligned} \alpha^{PLA} &= \frac{k^3 - k}{s^3 - s}.\alpha_1^{PLA} + \frac{(s-k)^3 - (s-k)}{s^3 - s}.\alpha_2^{PLA} \\ &+ \frac{3k(s-k)}{s^2 - 1}.\alpha^{MITH} \end{aligned}$$

5.2 Updating the Error Rate

THEOREM 1. *The error of MITH on segment $S = S_1 \cup S_2$, approximating all the original data points, can be calculated recursively from the weighted errors of segments S_1 and S_2 and the slope α of segment S.*

Proof

The sum-of-squares error of S can be decomposed into two parts: $E(S) = \sum_{i=1}^{k}(j_i - \alpha.i - \beta)^2 + \sum_{i=k+1}^{s}(j_i - \alpha.i - \beta)^2$. The first part of this decomposition can be rewritten as follows: $\sum_{i=1}^{k}(j_i - \alpha.i - \beta - (\alpha_1.i + \beta_1) + (\alpha_1.i + \beta_1))^2 = \sum_{i=1}^{k}(j_i - \alpha_1.i - \beta_1)^2 + (\alpha_1.i + \beta_1 - \alpha.i - \beta)^2 + 2(j_i - \alpha_1.i - \beta_1)\big((\alpha_1 - \alpha).i + (\beta_1 - \beta)\big)$

The first sum of squared values in the first line of the previous rewriting, represents the sum-of-squares error of S_1 with MITH. The second part represents the error between the MITH line of segment S_1 and the MITH line of the segment S computed on segment S_1. Like the PLA model, MITH verifies that $\sum_{i=1}^{k}(j_i - \alpha_1.i - \beta_1) = 0$ and the last part must be simplified by: $\Delta(S_1/\alpha) = 2(\alpha_1 - \alpha)\sum_{i=1}^{k}(j_i - \alpha_1.i - \beta_1)i = 2(\alpha_1 - \alpha)\sum_{i=1}^{k} i.j_i - k(\alpha_1 - \alpha)\big(\alpha_1(2k+1)(k+1)/3 - \beta_1(k+1)\big)$

$E(S)$ must be written as follows: $E(S) = E(S_1) + E(S_2) + \hat{E}(S_1 \cup S_2) + \Delta(S_1/\alpha) + \Delta(S_2/\alpha)$. The computation of $\Delta(S_1/\alpha)$ or $\Delta(S_2/\alpha)$ depends on the slope of the new segment S and the values $\delta(S_1) = \sum_{i=1}^{k} i.j_i$ and $\delta(S_2) = \sum_{i=k+1}^{s} i.j_i$. Therefore a linear relation $\delta(S) = \delta(S_1) + \delta(S_2)$ exists and δ is maintained for each segment. Furthermore, $\Delta(S_1/\alpha)$ can be easily computed as $\Delta(S_1/\alpha) = -\alpha.\delta(S_1) + \Lambda(S_1)$ where $\Lambda(S_1)$ depends only on the segment S_1 and we obtain: $E(S) = (E(S_1) + \Lambda(S_1)) + (E(S_2) + \Lambda(S_1)) + \hat{E}(S_1 \cup S_2) - \alpha(\delta(S_1) + \delta(S_2))$

6. EXPERIMENTS

This section is intended to evaluate the efficiency of our approach in a series of experiments. We have implemented GEAR with both PLA and MITH representations. We have also implemented a wavelet algorithm and a TTW algorithm in order to compare GEAR to the most important existing techniques. A description of the TTW technique can be found in [4]. Though many time series datasets are available for download, they generally contain only one series for each dataset. Actually, it is very difficult to find one dataset made of a large number of time series corresponding to the same "system". Furthermore, it is not reasonable to mix series related to "weather", "finance" and "space observation" (for instance) in order to simulate a dataset of multiple time series. In fact, we need to work on calibrated datasets having multiple time series (for instance the values of 100 stocks on several years). We evaluated our algorithms on two real-world datasets. The first dataset (NYSE in the rest of this section) has been built thanks to data downloaded from `http://icf.som.yale.edu/nyse`. It contains 134 times series, each made of 804 values corresponding to the price and dividend information on NYSE stocks from 1815 to 1925. The second dataset (WEB in the rest of this section) comes from the Web access logs of *Anonymized Lab*. We have collected one year of usage in these files for a total of 14 Gb. A first preprocessing aimed to extract the top frequent URLs from this dataset. We found 223 URLs, with a minimum support of 1%. Then, we have reported in our WEB dataset the fluctuations of the requests on these pages with a jumping window of size 1000. In other words, each time series is updated every 1000 requests and the new value corresponds to the number of requests to the URL represented by that time series. To summarize, we have tested our method on datasets corresponding to 134 and 223 time series (357 time series in total).

GEAR (MITH and PLA versions) are implemented in Java. The experiments are performed on a standard machine equipped with 2 Gb memory and a Pentium 4 processor operating at 2.2 Ghz. *Our algorithms and datasets are available on request and all our experiments can be repeated.*

6.1 Comparing MITH to PLA

In this first set of experiments we evaluate, on both datasets:

- the impact of MITH on the average error at each step;
- the difference in terms of execution between the MITH and PLA versions of GEAR.

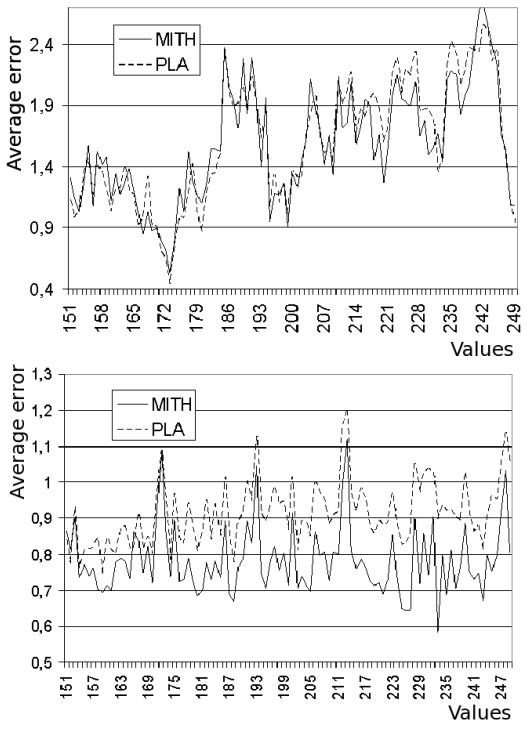

Figure 2: Step by step average error of MITH and PLA on NYSE (top) and WEB (bottom) datasets.

Figure 2 shows a sample of the average error of MITH and PLA in the GEAR framework for NYSE (top) and WEB (bottom) datasets. The error is computed as follows: $\forall i \in [1..s] err[i] = averageError_{j=1}^{j=n}(R[j][i])$ where $R[j][i]$ is the error (absolute value) between the representation of the i^{th} record in time series j and its real value in the dataset. In other words, we measure the error in $\mathcal{L}_1$. Obviously, in $\mathcal{L}_2$ the error of MITH cannot be better than that of PLA. On the other hand, we consider the $\mathcal{L}_1$ error as very important since it is less sensitive to outliers. Therefore, for each time unit, we know the average error of a representation for the

M	Mith/NYSE	RL/NYSE	Mith/WEB	RL/WEB
4000	2646	2863	4729	5652
4500	2756	2998	4480	5793
5000	2574	3000	4800	5906
5500	2637	3063	4624	5965
6000	2714	3049	5082	6024
6500	2697	2933	5069	6229

Table 1: Execution times of MITH and PLA on NYSE and WEB datasets with varying number of segments (M).

n time series at that time. For NYSE, the average errors of MITH and PLA are very similar. The effectiveness of MITH is illustrated on the WEB dataset. For this dataset MITH has an excellent average error compared to that of PLA. Subsection 6.2 gives the average global errors of both representations and MITH allows GEAR to obtain a very low global error (0.88) compared to PLA (2.53). This is due to the fact that MITH is not intended to minimize the sum squared error. therefore, our goal to lower the $\mathcal{L}_1$ error is better achieved by MITH (as illustrated by Figure 2).

Table 1 shows the execution times of GEAR when utilizing MITH and PLA for both datasets. The response times of MITH are systematically lower than those of PLA. The explanation simply relies on the mere formulas used by MITH (given in Section 5) compared to the complex formula of traditional PLA.

6.2 Comparing GEAR to TTW and Wavelets

In this section, our goal is to compare the error of GEAR with those of TTW and wavelets. Table 2 gives the global average error of each representation, compared to the original data. For each representation the average error at each step is computed as described in subsection 6.1. Then we compute the mean value for the average errors vector. Each representation (MITH and PLA) is given a total number of 5000 segments to approximate the time series. The representation by means of TTW requires 11 segments for NYSE and for WEB (this number is given by the logarithm of s, the number of values for each time series, in the dataset). The representation by means of Haar wavelets is given a number of segments corresponding to that of GEAR (*i.e.* M segments that are divided into M/n coefficients given to each time series, with n the number of time series).

Table 2: Global errors of TTW, Wavelets and GEAR with PLA and MITH

	WEB	NYSE		WEB	NYSE
TTW	17.24	2.45	PLA	2.53	0.86
Wavelets	14.05	1.56	MITH	0.88	0.87

7. CONCLUSION

In this paper we have presented (1) GEAR, a framework for summarizing streaming time series and (2) MITH an innovative representation of time series allowing fast computing. GEAR is able to manage numerous time series of any length thanks to its error balancing principle. This principle allows each representation of a time series to obtain or to give back segments. MITH is compared to PLA both with formal and experimental studies. Our experiments assess the relevance of this proposal. First, GEAR allows a better handling of multiple streaming time series by lowering the average global error. Second, approximation algorithm based on MITH has fast response times compared to the same algorithm based on PLA. Furthermore, in our experiments, MITH shows a gain in precision when measuring the global error in $\mathcal{L}_1$. Possible future tracks include 1) the study of a fast error approximation based on surfaces and the utilization of MITH for queries and 2) to study the data configurations where the precision of MITH is better than PLA.

8. REFERENCES

[1] E. Airoldi and C. Faloutsos. Recovering latent time-series from their observed sums: network tomography with particle filters. In *KDD'04*, pages 30–39, New York, NY, USA, 2004. ACM.

[2] Y.-A. L. Borgne, S. Santini, and G. Bontempi. Adaptive model selection for time series prediction in wireless sensor networks. *Signal Process.*, 87(12):3010–3020, 2007.

[3] J. H. Chang and W. S. Lee. Finding recent frequent itemsets adaptively over online data streams. In *KDD'03*, pages 487–492, 2003.

[4] Y. Chen, G. Dong, J. Han, B. Wah, and J. Wang. Multidimensional regression analysis of time-series data streams. In *VLDB*, 2002.

[5] H. Cheng and P.-N. Tan. Semi-supervised learning with data calibration for long-term time series forecasting. In *KDD'08*, pages 133–141, New York, NY, USA, 2008. ACM.

[6] E. Cohen and M. Strauss. Maintaining time-decaying stream aggregates. In *PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 223–233, 2003.

[7] J. Lin, E. J. Keogh, S. Lonardi, and B. Y. chi Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD*, pages 2–11, 2003.

[8] T. Palpanas, M. Vlachos, E. J. Keogh, and D. Gunopulos. Streaming time series summarization using user-defined amnesic functions. *IEEE Trans. Knowl. Data Eng.*, 20(7):992–1006, 2008.

[9] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, pages 697–708, 2005.

[10] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *VLDB*, pages 385–394, 2000.

[11] Y. Zhao. Generalized dimension-reduction framework for recent-biased time series analysis. *IEEE Trans. on Knowl. and Data Eng.*, 18(2):231–244, 2006. Senior Member-Shichao Zhang.

[12] Y. Zhu and D. Shasha. Statstream: statistical monitoring of thousands of data streams in real time. In *VLDB'02*, pages 358–369. VLDB Endowment, 2002.