Continuous Trend-Based Classification of Streaming Time Series

Maria Kontaki*, Apostolos N. Papadopoulos, and Yannis Manolopoulos

Department of Informatics, Aristotle University,
GR-54124 Thessaloniki, Greece
{kontaki, apostol, manolopo}@delab.csd.auth.gr

Abstract. Trend analysis of time series data is an important research direction. In streaming time series the problem is more challenging, taking into account the fact that new values arrive for the series, probably in very high rates. Therefore, effective and efficient methods are required in order to classify a streaming time series based on its trend. Since new values are continuously arrive for each stream, the classification is performed by means of a sliding window which focuses on the last values of each stream. Each streaming time series is transformed to a vector by means of a Piecewise Linear Approximation (PLA) technique. The PLA vector is a sequence of symbols denoting the trend of the series (either UP or DOWN), and it is constructed incrementally. Efficient in-memory methods are used in order to: 1) determine the class of each streaming time series and 2) determine the streaming time series that comprise a specific trend class. Performance evaluation based on real-life datasets is performed, which shows the efficiency of the proposed approach both with respect to classification time and storage requirements. The proposed method can be used in order to continuously classify a set of streaming time series according to their trends, to monitor the behavior of a set of streams and to monitor the contents of a set of trend classes.

Keywords: data streams, time series, trend detection, classification, data mining.

1 Introduction

The study of query processing and data mining techniques for data stream processing has recently attracted the interest of the research community [2], due to the fact that many applications deal with data that change very frequently with respect to time. Examples of such application domains are network monitoring, financial data analysis, sensor networks to name a few. The most important property of data streams is that new values are continuously arrive, and therefore efficient storage and processing techniques are required to cope with the high update rate.

* This research is supported by the *State Scholarships Foundation (I.K.Y.).*

J. Eder et al. (Eds.): ADBIS 2005, LNCS 3631, pp. 294–308, 2005.

A streaming time series S is a sequence of real values s_1, s_2, ..., where new values are continuously appended as time progresses. For example, a temperature sensor which monitors the environmental temperature every five minutes, produces a streaming time series of temperature values. As another example, consider a car equipped with a GPS device and a communication module, which transmits its position to a server every ten minutes. A streaming time series of two-dimensional points (the x and y coordinates of its position) is produced. Note that, in a streaming time series data values are ordered with respect to the arrival time. New values are appended at the end of the series.

A class of algorithms for stream processing focuses on the recent past of data streams by applying a *sliding window* on the data stream [2,3]. In this way, only the last W values of each streaming time series is considered for query processing, whereas older values are considered obsolete and they are not taken into account. As it is illustrated in Figure 1, streams that are non-similar for a window of length W (left), may be similar if the window is shifted in the time axis (right).

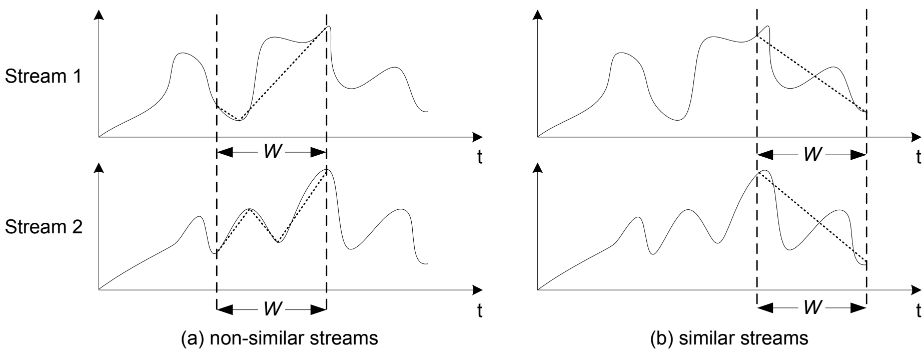

Fig. 1. Similarity using a sliding window of length W

We use trends as a base to classify streaming time series for two reasons. First, *trend* is an important characteristic of a streaming time series. In several applications the way that stream values are modified is considered important, since useful conclusions can be drawn. For example, in a stock data monitoring system it is important to know which stocks have an increasing trend and which ones have a decreasing trend. Second, trend-based representation of time series is more close to the human intuition. In the literature, many papers [6,7] use the values of the data streams and a distance function like Euclidean distance to cluster streams. Although a distance can be large for a pair of streams, these two streams can be intuitionally considered similar, if their plots are examined. Thus, distance functions aren't always good metrics to cluster or to classify objects.

In this paper, we focus on the problem of continuous time series classification based on the trends of the series as time progresses. Evidently, we expect that the same series will show different trend for different time intervals. The classification is performed by considering the last W values of each stream (in a sliding window manner). Again, two streaming time series that show similar trends for a specific time interval may be totally dissimilar for another time interval. This effect is illustrated in Figure 1, where the trends of the time series are represented by dotted lines. We note also that two series which show similar trends may be completely different with respect to the values they assume.

The rest of the article is organized as follows. In Section 2 we give significant related work on the issue of trend analysis in streams. Section 3 discusses in detail the proposed approach which is based on two important issues: 1) an effective in-memory representation of the streams by means of an approximation and 2) an efficient in-memory organization in order to quickly categorize a stream when new values for that stream are available. Experimental results based on real-life datasets are offered in Section 4, whereas Section 5 concludes the work and raises some issues for further research in the area.

2 Related Work and Contribution

The last decade, mining time series has attracted the interest of the researchers. Classification is a well-known data mining problem. Many papers have been proposed to classify objects from different research domains as machine learning, knowledge discovery and artificial intelligence.

The classification problem is more challenging in the case of streaming time series due to the dynamic nature of the streaming case. In the recent past, [1] proposed a classification system in which the training model adapts to the changes of the data streams. The method is based on the micro-clusters, vectors which contain simple statistics over a time period of a stream. Classification is achieved by combining micro-clusters in different time instances (snapshots). The method uses a periodically scheme to update the micro-clusters and reports the classification on demand. Our method incrementally computes and continuously reports the classification. Moreover the scheme, that was used, needs a training set in opposition to our scheme that has a restricted number of classifiers and the classifiers are a priori known. In [12] used info-fuzzy networks to address the problem. Other approaches include one-pass mining algorithms [4,8], in which the classification model is constructed in the beginning, and therefore do not recognize possible changes in the underlying concept.

Piecewise linear approximation has been used to represent efficiently time series in many topics as clustering, classification and indexing [16,17]. Many variations have been proposed, among them are the piecewise aggregate approximation (PAA) [11] that stores the mean value of equal-length segments and the adaptive piecewise constant approximation (APCA) [10] that stores the mean value and the right end-point of variable-length segments.

Trend analysis has been used to cluster time series in many domains such as time series [18,13], bioinformatics [14] and ubiquitous computing [15]. Yoon et al proposed six trend indicators. A time series is represented as a partial order of the indicators. A bitmap index is used to encode indicators into bit strings in order to compute the distance between two time series with the XOR operator. In [13,14] modifications of PLA are used to detect trends and three types of them are used (up, down and steady) to cluster time series. These methods study the clustering problem in time series. They do not use an incremental way to compute the trend representation. Additionally the clustering algorithms were proposed are not one-pass algorithms. So the methods are not appropriate in a streaming case. In comparison with our method the trend representation is incrementally computed and the classification is continuously reported using an efficient in-memory access method. Recently, [16] proposed trend analysis to address the problem of subsequence matching in financial data streams. The Bollinger Band indicator (%b) is used to smooth time series and then the PLA is applied. The %b indicator uses simple moving average and thus the whole sliding window is required to compute next values of %b. So the pla representation is not computed incrementally and in case of thousand of streams the memory requisites are enormous.

The contribution of the work is summarized as follows:

- An incremental computation of the PLA approximation is presented, which enables the continuous representation of the time series trends under the sliding window paradigm.
- An efficient in-memory access method is proposed which facilitates fundamental operations such as: determine the class of a stream, insert a stream into another class, delete a stream from an existing class.
- Continuous trend-based classification is supported, which enables the monitoring trend classes or the monitoring of data stream.
- The proposed technique can be applied even in the case where only a subset of the data streams change their values at some time instance. Therefore, it is not required to have stream values at every time instance for all streams.

3 Trend Representation and Classification

In data stream processing there are two important requirements posed by the nature of the data. The first requirement states that processing must be very efficient in order to allow continuous processing due to the large number of updates. This suggests the use of the main memory in order to avoid costly I/O operations. The second requirement states that random access to past stream data is not supported. Therefore, any computations that must be performed on the stream should be incremental, in order to avoid reading past stream values. In order to be consistent with the previous requirements, we propose a continuous classification scheme which requires small storage overhead and performs the classification in an incremental manner, taking into consideration the synopsis of each stream. Each stream synopsis requires significantly less storage than the raw stream data, and therefore, better memory utilization is

Table 1. Basic notations used throughout the study

Symbol	Description
S	a streaming time series
$S(t)$	the value of stream S at time t
N	number of streaming time series
n	length of a streaming time series
W	sliding window length
p	period of moving average ($p \leq W$)
$EMAi_p(t)$	the i-th exponential moving average of period p ($t \geq p$)
$TRIX(t)$	percentage differences of $EMA3_p(t)$ signal
PLA	piecewise linear approximation
$PLA(i)$	the i-th segment of the PLA
k	the number of segments of the PLA
tl_{min}	the minimum time instance of a bucket list
tl_{max}	the maximum time instance of a bucket list
tb_{min}	the minimum time instance of a bucket
tb_{max}	the maximum time instance of a bucket

achieved. Before we describe the proposed method in detail we give the basic symbols used throughout the study in Table 1.

3.1 Time Series Synopsis

In this section we study the problem of the incremental determination of each stream synopsis, in order to reduce the required storage requirements and enable stream classification based on trend. Trend detection has been extensively studied in statistics and related disciplines [5,9]. In fact, there are several indices that can be used in order to determine trend in a time series. Among the various approaches we choose to use the TRIX indicator [9] which is computed by means of a triple moving average on the raw stream data. We note that before trend analysis is performed, a smoothing process should be applied towards removing noise and producing a smoother curve, revealing the time series trend for a specific time interval. This smoothing is facilitated by means of the TRIX indicator, which is based on a triple exponential moving average calculation of the logarithm of the time series values. In the sequel, we first explain the use of the exponential moving average and then we introduce the TRIX indicator.

Definition 1.
The exponential moving average of period p over a streaming time series S is calculated by means of the following formula:

$$EMA_p(t) = EMA_p(t-1) + \frac{2}{1+p} \cdot (S(t) - EMA_p(t-1)) \quad (1)$$

Definition 2.
The TRIX indicator of period p over a streaming time series S is calculated by means of the following formula:

$$TRIX(t) = 100 \cdot \frac{EMA3_p(t) - EMA3_p(t-1)}{EMA3_p(t-1)} \tag{2}$$

where EMA3$_p$ is a signal generated by the application of a triple exponential moving average of the input time series.

The signal $TRIX(t)$ oscillates around the zero line. Whenever $TRIX(t)$ crosses the zero line, it is an indication of trend change. This is exactly what we need in order to perform a trend representation of an input time series. Figure 2 illustrates an example. Note that the zero line is crossed by the $TRIX(t)$ signal, whenever there is a trend change in the input signal. Figure 2 also depicts the smoothing achieved by the application of the exponential moving average.

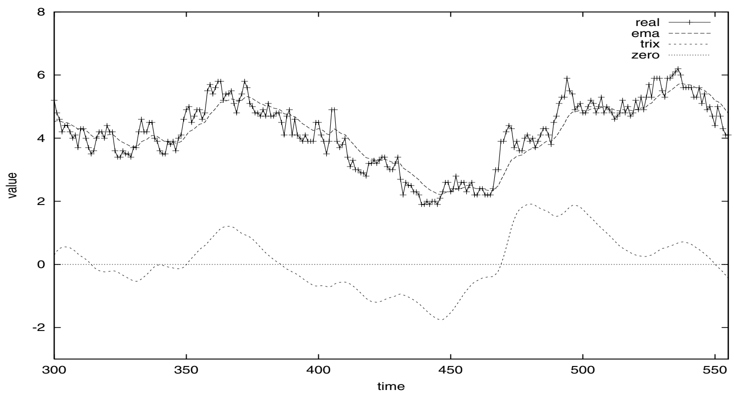

Fig. 2. Example of a time series and the corresponding $TRIX(t)$ signal

Definition 3.
The PLA representation of a streaming time series S for a time interval of W values is a sequence of at most W-1 pairs of the form $(t, trend)$, where t defines the left-point time of the segment and *trend* denotes the trend of the stream (UP or DOWN) in the specified segment.

Each time a new value arrives, the PLA is updated. Three operations (ADD, UPDATE, EXPIRE) are implemented to support incremental computation of the PLA. The ADD operation is applied when a trend change detected and adds a new PLA-point. The UPDATE operation is applied when the trend is stable and updates the timestamp of the last PLA-point. The EXPIRE operation is applied when the first segment of the PLA is expired and deletes the first PLA-point. Notice that when the UPDATE operation is applied the class of the stream does not change.

3.2 Continuous Classification

In this section we study the way continuous classification is performed. Taking into account that each PLA segment has an UP or DOWN direction, the number of possible trend classes for a sliding window of length W is given by $C_W = 2 \cdot (W - 1)$ as it is illustrated by the following proposition.

Proposition.
The number of different classes C_W of streaming time series is given by:

$$C_W = 2 \cdot (W - 1) \tag{3}$$

where W is the sliding window length.

Proof.
To prove this proposition we use induction. Evidently, the proposition is true for W=2 (note that W=2 is the smallest value for the sliding window length which enables trend determination). We assume that the proposition is true for W=n, and therefore $C_n = 2 \cdot (n\text{-}1)$. We will prove the proposition for W=n+1. The values at positions n and n+1 define a straight line with either an increasing trend (UP) or a decreasing trend (DOWN) (in the case where the TRIX indicator is zero, we retain the previous trend). If the trend is UP and the trend of the previous PLA segment is also UP, then the final result is UP. If the trend is DOWN and the trend of the previous PLA segment is also DOWN, then the final result is DOWN. If one of the above cases is true, then the (n+1)-th stream value has no contribution at all. Now consider the case where the last trend is UP and the previous trend is DOWN, or the case where the last trend is DOWN and the previous trend is UP. If one of the aforementioned cases is true then clearly, the (n+1)-th stream value contributes to another trend class. This means that the (n+1)-th stream value can give two more trend classes. This means that $C_{n+1} = C_n + 2$. By the induction hypothesis we know that $C_n = 2 \cdot (n\text{-}1)$. Therefore, $C_{n+1} = 2 \cdot (n\text{-}1) + 2 = 2 \cdot n$, and this completes the proof. □

Every time a new value for a streaming time series arrives, the corresponding stream may change from a trend class to another. We illustrate the way continuous classification can be achieved efficiently, by means of an in-memory access method which organizes the streams according to the trend class they belong and by taking into account time information to facilitate efficient search. During continuous classification the following operations must be supported:

- We must quickly locate the class that the corresponding stream belongs to,
- We must delete (if necessary) the corresponding stream from the old class and assign it to a new one, and
- We must report efficiently the stream identifiers that belong to a specific trend class.

Each trend class is supported by several lists of buckets. The first bucket of each list is the *primary bucket* whereas the other buckets are *overflow buckets*. The *overflow buckets* are used only in the case where the stream must be inserted in an existing list (step 2 of Algorithm Insert) and the *primary bucket* of the list is full (bucket size exceeded). Each bucket list is characterized by two time instances tl_{min} and tl_{max}, denoting the minimum and the maximum time instances which corresponds to the $k-1$-th PLA point, where k is the number of points contained in the PLA representation. We use the one before the last PLA point as base to insert streams in bucket lists because is the last stable point (the last point maybe changed if an update happens) and thus we have to update the classification structure only when the stream changes class. Each bucket is composed of a set of stream identifiers and two time instances tb_{min} and tb_{max}. These time instances denote the time interval that each stream in the bucket has been inserted.

In Figure 3 an example of the structure is depicted. The class DUD consists of two bucket lists. The first list contains additionally an *overflow bucket*. For the first list the tl_{min} is 10 and the tl_{max} is 15. This means that the streams 1,2,5,8 have the one before the last PLA point between time 10 and 15. For the *primary bucket* of the first list the tb_{min} is 12 and the tb_{max} is 17 and contains the streams 2,5 and 8. Therefore streams 2,5 and 8 were inserted in this class between time 12 and 17. For the *overflow bucket* of the first list the tb_{min} is 18 and the tb_{max} is 18 and contains the stream 1. Stream 1 was inserted at time instance 18. The description of the second list is the same.

We will explain how we use the bucket lists structure to continuous classify streams with an example. Assume the two bucket lists of the classes DUD and DUDU of the Figure 3. The bucket size is 3 and the window size is 16. At time instance 21 a new value for the stream 1 is arrived. The following operations take place: a) we search the stream 1 in the bucket lists of class DUD, b) we delete it, c) we update PLA and d) we insert it in the bucket lists of class DUDU. The stream 1 has the one before last PLA point at time 14. We search for the bucket list in which tl_{min} and tl_{max} enclose time 14 (step 1 of search algorithm). This is the first list. The first list contains an *overflow bucket* so we must find the insertion time of the stream 1 (insertion time algorithm). The stream 1 was inserted in this class either when a new PLA-point was added ($PLA(k-1)$-point + 1) or when the first segment expired (W + $PLA(0)$-point - 1). The maximum of these two times is the time that the stream was inserted. Therefore the insertion time is 18. We search in the list, a bucket in which tb_{min} and tb_{max} enclose time 18 (step 3 of search algorithm). This is the *overflow bucket* (figure 8). We delete stream 1 and then we delete the bucket because is empty (delete algorithm). Then we update the PLA of the stream. The new class is the DUDU class. Now the one before the last PLA point is at time 20. Since the bucket lists of this class is not empty (step 1 of the insert algorithm) and since the tl_{max} of the one before the last bucket list is smaller than 20 (step 2), we check if the last bucket list is full (step 3). In the Figure 9 we can see that the *primary bucket* of this list is not full. So we update the tl_{max} (step 3) and the tb_{max} and we insert the stream in the *primary bucket* of this list (step 5). The algorithms for insert, search and delete are given in Figure 4, Figure 5 and Figure 7 respectively.

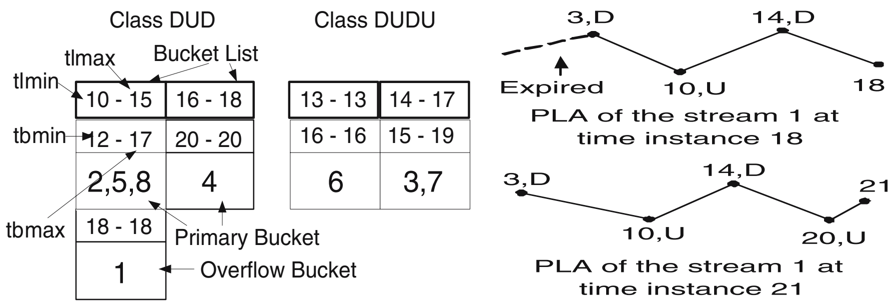

Fig. 3. Example of search algorithm with bucket size 3

Algorithm. Insert

/* Determine the list to insert the stream */
1. If the corresponding class is empty, then a new list is created and the values tl_{min} and tl_{max} are set to the time instance t_{n-1} of the $(n-1)$-th PLA point.
2. Otherwise, check if t_{n-1} is less than the tl_{max} value of the last list. If yes, then the stream identifier is inserted into one of the existing bucket lists. The appropriate bucket list is the list in which the tl_{min} and tl_{max} enclose the t_{n-1}.
3. Otherwise, check if the primary bucket of the last list is full. If the primary bucket is not full then the stream is inserted into that list by updating the corresponding value tl_{max}. If the primary bucket is full, a new bucket list is generated and the values tl_{min} and tl_{max} are set to the time instance t_{n-1} of the $(n-1)$-th PLA point.

/* Determine the bucket to insert the stream */
4. If the primary bucket of the current list does not exist, then a primary bucket is created and the stream is inserted. The tb_{min} and tb_{max} values are updated with the current time.
5. If the primary bucket of the current list is not full, then the stream is inserted into that bucket and the tb_{max} value is updated with the current time.
6. Otherwise the stream is inserted into the last overflow bucket of the list, by updating accordingly the tb_{max} value. If the last overflow bucket is full, a new overflow bucket is generated.

Fig. 4. Insertion algorithm

Algorithm. Search

---

1. Determine the bucket list by checking for the values of tl_{min} and tl_{max} that enclose the time instance t_{n-1} of the stream.
2. If the list contains only a primary bucket, then the stream identifier is found into that bucket.
3. If the list contains a number of overflow buckets, then by using the time instance that the stream has been inserted (Fig. 6), the corresponding overflow bucket which contains the stream is easily detected.

Fig. 5. Search algorithm

4 Performance Study

The proposed trend-based classification scheme has been implemented in C++, and the experimental evaluation has been performed on a Pentium IV machine

Algorithm. Insertion Time

1. Compute the time that the last expiration has occurred. The time is given by $lastEXP{=}W + PLA(0)$-point - 1.
2. Compute the time that the last ADD operation has occurred. The time is given by $lastADD{=} PLA(k-1)$-point + 1.
3. The time that the stream has been inserted is given by max($lastEXP$,$lastADD$).

Fig. 6. Insertion Time algorithm

Algorithm. Delete

1. Call algorithm Search in order to determine the position of the stream.
2. Remove the stream identifier from the bucket.
3. If the bucket is empty it is removed.
4. If the bucket list is empty it is removed.

Fig. 7. Deletion algorithm

with 1GByte RAM running Windows 2000. Two real-life datasets with different characteristics have been used:

- **STOCKS**: is the daily stock prices obtained from http://finance.yahoo.com. The data set consists of 93 time sequences, and the maximum length of each one is set to 3,000.
- **TAO**: this dataset (Tropical Atmosphere Ocean) contains the wind speed of 65 sites on Pacific and Atlantic Ocean since 1974, obtained from the Pacific Marine Environmental Laboratory (http://www.pmal.noaa.gov/tao). We have used the highest data resolution (e.g. the sampling time interval) that was available. About 12,000 streams form the data set, and the maximum length of each one is set to 1,000.

In the sequel we give the performance results for different parameter values for the sliding window length (W), the exponential moving average period (p), the number of the streaming time series (N), the bucket size(B). The experiments are divided into two categories. The first category studies the quality of the clustering and the second studies the performance. We focus on two performance measures: the computational cost required to perform continuous classification and the memory requirements of the proposed approach because they are the most important metrics in determining the effectiveness and the robustness of a stream processing system. The CPU cost was measured in seconds. Finally, the proposed method works both in cases where all the streams or part of them are updated. For the experiments below, the first case was used.

4.1 Quality of PLA

The underlying idea of the approach is to cluster streams using an abstractive representation of the streams that is closer to the "human sense" despite using the values of the streams and the Euclidean distance or others distance metrics. In this section we examined the conforming between the piecewise linear approximation of a stream and the general shape of a stream without micro changes.

Next, we give some classification examples. Figure 8 shows classification patterns and a sample of streams that are associated with each one. For each stream, both the raw data and the PLA are illustrated. The classification instances are peaked after a random number of updates. Notice that if we are not contented with the representation, we can choose a greater p for a more abstractive description of the stream, or a smaller p for a more comprehensive description.

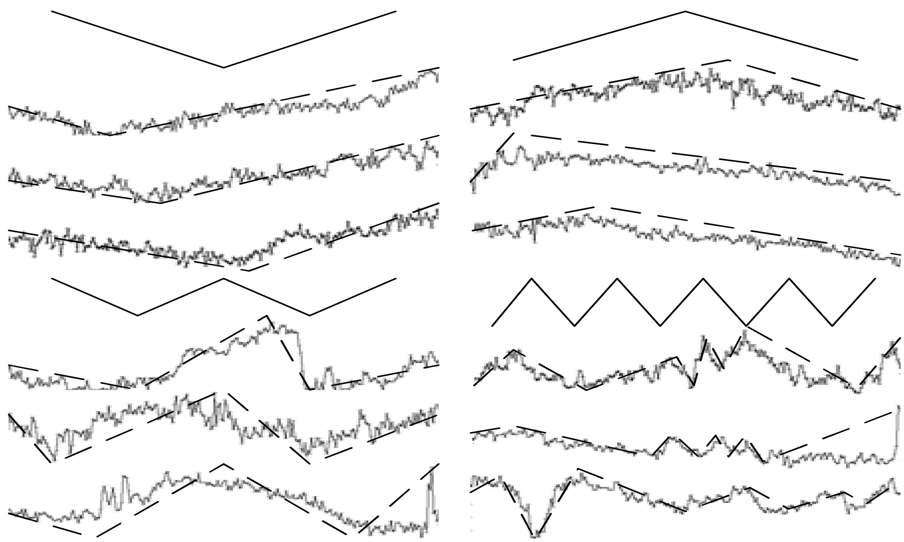

Fig. 8. Classification examples

Additionally, Figure 9 shows the number of clusters for different values of p with respect to W for the TAO and STOCKS data sets. The term CL_raw is used for the possible number of clusters that is entirely depended on the window size W. It was expected the number of clusters, that is actually used, is reduced as the p is increased because less details are represented by the PLA. Therefore some streams are moved in classes with smaller number of segments.

4.2 Performance Evaluation

We first examine the performance of the method with respect to window length. Figure 10 illustrates the total CPU cost (10a) and the CPU cost to compute the PLA of all streams for all the updates (10b) for the TAO data set. Different values for p are used. From Figure 10, the total CPU cost is determined from

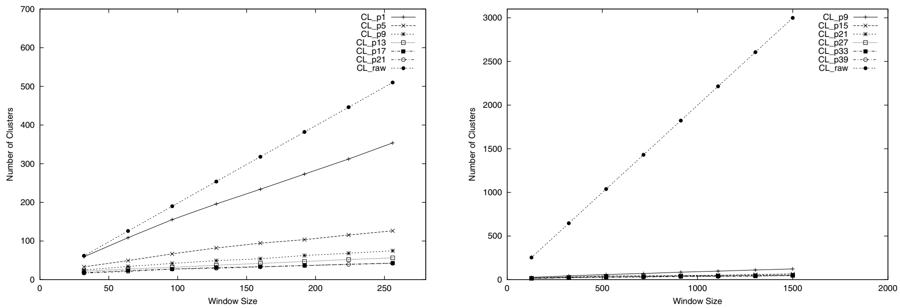

Fig. 9. Number of clusters vs window length for a) TAO and b) STOCKS data sets

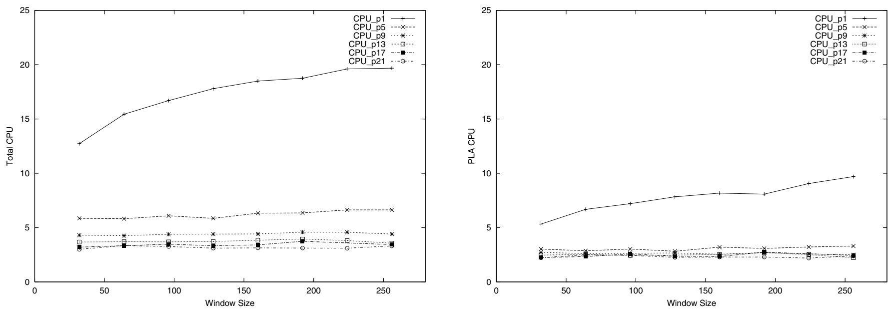

Fig. 10. a) Total CPU cost and b) PLA CPU cost vs window length

the PLA CPU cost. The latter is independent from the window size due to the use of the TRIX indicator.

Table 2 illustrates the total memory for the STOCKS data set and partial memory prerequisites for the PLA representation and the classification structure. Total memory is essentially affected by the PLA memory. The PLA memory is increased as the window size increases.

Next we examine the performance of our method with respect to the number of streams. Figure 11a depicts the CPU cost for all the streams (12145) and for all the updates (about 700) for the TAO data set. The term TOTAL_CPU is used for the sum of the PLA and the classification CPU cost. The CPU cost increases linearly with respect to the number of streams.

The memory prerequisites of the PLA per update for the TAO data set are illustrated in Figure 11b. The term MEM_raw is used for the memory prerequisites of the raw data. Notice that the y-axis scales logarithmically. The PLA memory increases steadily with respect to the number of streams but it is less than the 10% of raw data memory.

To better understand the influence of the bucket size in the classification method, Figure 3 shows the CPU cost and the memory prerequisites of the classification method. Large bucket size reduces the memory prerequisites but increases CPU cost, whereas a small bucket size has the opposite results. The bucket size is a trade-off between memory resources and computation time.

Table 2. Total CPU and classification memory vs bucket size

Window Size	Total memory (KB)	Classification memory (%)	PLA memory (%)
128	13013.797	28.6%	71.4%
324	16065.762	25.9%	74.1%
520	19059.859	23.9%	76.1%
716	21772.871	21.4%	78.6%
912	24441.957	19.6%	80.4%
1108	27129.715	18.1%	81.9%
1304	29934.621	17.2%	82.8%
1500	32726.527	16.5%	83.5%

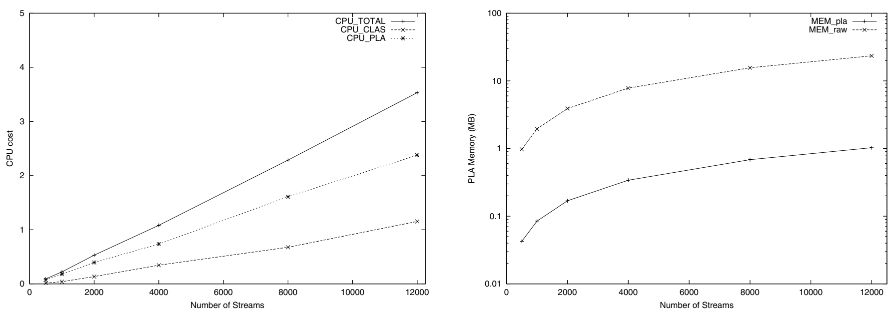

Fig. 11. a) CPU cost and b) memory prerequisites of PLA vs number of streams for TAO

Table 3. Total CPU and classification memory vs bucket size

Bucket Size	Total CPU	Classification memory (MB)
50	3.745	25.061
100	3.7842	14.803
200	3.6836	8.573
300	3.73	6.082
400	3.801	4.764
500	3.9377	3.876
600	4.0029	3.286

5 Conclusions and Future Work

Trend analysis of time evolving data streams is a challenging problem due to the fact that the trend of a time series changes with respect to time. In this paper we studied the problem of continuous trend-based classification of streaming time series, by using a compact representation for each stream and an in-memory access method to facilitate efficient search, insert and delete operations. A piecewise linear approximation (PLA) has been used in order to determine the trend curve of each stream. The PLA representation has been applied on a smoothed version of each stream. We have used the TRIX indicator for smoothing. Moreover, a continuous classification method has been presented which reassigns a stream to new trend class if necessary. Performance evaluation results based on real-life datasets have shown the feasibility and the efficiency of the proposed approach.

In the near future we plan to extend the current work towards continuous clustering of streaming time series, by taking into account the similarity between trend classes.

References

1. Aggarwal, C. C., Han, J., Yu, P. S.: On Demand Classification of Data Streams, Proceedings of the International Conference of Knowledge Discovery and Data Mining(KDD), WA, USA (2004)
2. Babcock, B. Babu, S., Datar, M., Motwani, R., Widom, J.: Models and Issues in Data Stream Systems, Proceedings ACM PODS, Madison, Wisconsin (2002) 1–16
3. Datar, M., Gionis, A., Indyk, P., Motwani, R.: Maintaining stream statistics over sliding windows, Proceedings of the 2002 Annual ACM-SIAM Symp. on Discrete Algorithms (2002) 635–644
4. Domingos, P., Hulten, G.: Mining High-Speed Data Streams, Proceedings of ACM SIGKDD Conference (2000)
5. Fung, G. P. C., Yu, J. X., Lam, W.: News Sensitive Stock Trend Prediction, In PAKDD (2002) 481-493
6. Guha, S., Meyerson, A., Mishra, N., Motwani, R., OCallaghan, L.: Clustering Data Streams: Theory and Practic, IEEE TKDE, Vol. 15, No. 3 (2003) 515-528
7. Guha, S., Mishra, N., Motwani, R.., OÆCallaghan, L.: Clustering data streams, In Proc. of the 2000 Annual IEEE Symp. on Foundations of Computer Science (2000) 359–366
8. Hulten, G., Spencer, L., Domingos. P.: Mining Time Changing Data Streams, Proceedings of ACM KDD Conference (2001)
9. Hutson, J. K.: TRIX - Triple Exponential Smoothing Oscillator, Technical Analysis of Stocks and Commodities (1983) 105–108
10. Keogh, E., Chakrabarti, K., Mehrotra, S., Pazzani, M.: Locally Dimensionality Reduction for Indexing Large Time Series Databases, Proceedings of ACM SIGMOD Conference, California, USA (2001)
11. Keogh, E., Pazzani, M.: A simple dimensionality reduction technique for fast similarity search in large time series databases, Proceedings of Pacific- Asia Conf. on Knowledge Discovery and Data Mining (2000) 122-133
12. Last, M.: Online Classification of Nonstationary Data Streams, Intelligent Data Analysis, Vol. 6, No. 2 (2002) 129-147
13. Ljubic, P., Todorovski, L., Lavrac, N., Bullas, J. C.: Time-series analysis of UK traffic accident data, Proceedings of the Conference on Data Mining and WareHouses (SiKDD), Ljubljana, Slovenia (2002)

14. Sacchi, L., Bellazzi, R., Larizza, C., Magni, P., Curk, T., U. Petrovic, U., Zupan, B.: Clustering and Classifying Gene Expressions Data through Temporal Abstractions, Proceedings of 8th Intelligence Data Analysis in Medicine and Pharmacology Workshop(IDAMAP 2003), Protaras, Cyprus (2003)
15. Takada, T., Kurihara, S., Hirotsu, T., Sugawara, T.: Proximity Mining: Finding Proximity using sensor Data History, Proceedings of 5th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), CA, USA (2003)
16. Wu, H., Salzberg, B., Zhang, D.: Online Event-driven Subsequence Matching over Financial Data Streams, Proceedings of ACM SIGMOD Conference, Paris, France (2004)
17. Yi, B.-K., Faloutsos, C.: Fast Time Sequence Indexing for Arbitrary Lp Norms, Proceedings of 26th International Conference on Very Large Databases (VLDB), Cairo, Egypt (2000)
18. Yoon, J. P., Luo, Y., Nam, J.: A Bitmap Approach to Trend Clustering for Prediction in Time-Series Databases, Proceedings of Data Mining and Knowledge Discovery: Theory, Tools, and Technology II, Florida, USA (2001)