Alternate Data Clustering for Fast Pattern Matching in Stream Time Series Data

Vishwanath R.H.[1], Thanagamani M.[1], Venugopal K.R.[1], Iyengar S.S.[2], and L.M. Patnaik[3]

[1] Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore University, Bangalore-560 001
[2] Florida International University, USA
[3] Indian Institute of Science, Bangalore
vishwanath_hulipalled@rediffmail.com

Abstract. Stream time series retrieval has been a major area of study due to its vast application in various fields like weather forecasting, multimedia data retrieval and huge data analysis. Presently, there is a demand for stream data processing, high speed searching and quick response. In this paper, we use a alternate data cluster or segment mean method for stream time series data, where the data is pruned with a computational cost of O(log *w*). This approach can be used for both static and dynamic stream data processing. The results obtained are the better than the existing algorithms.

Keywords: Stream time series, Alternate data clustering, Fast pattern match, Cluster mean.

1 Introduction

Time series is a sequence of values or events obtained over a repeated measurements of time. The values are typically measured at equal intervals of time (e.g., monthly, quarterly, half yearly, yearly, etc.,). In traditional time series, data processing, regular or frequent updates of the data is the most difficult task. Hence there is a necessity of new approach in the management of Stream time-series data.

A Stream time – series is continuously generated massive stream of data which is ordered and rigorously changing. Stream time-series is used in large spectrum of applications, such as sensor networks, electric power grids, moving object search, financial data analysis, internet traffic analysis and etc,. The dynamic environments where tremendous and potentially infinite volumes of data streams are required for incremental methods that update changing models, to identify and manage patterns of time series.

Motivation: Presently, there is demand for high speed processing of massive stream data leading to more efficient and accurate processing of data patterns which helps in real time applications like predicting a storm in advance in weather warning system,

V.V. Das and J. Stephen (Eds.): CNC 2012, LNICST 108, pp. 153–158, 2012.

to analyze the patterns in stock data monitoring system etc,. Previous works on similarity search on archived time series are not applicable to that over stream time series, since they cannot efficiently process frequent updates. Moreover, the existing work for searching on stream time-series data consider detecting a single static pattern over multiple stream time-series data [1], or checking which pattern (from multiple static patterns) is close to a single stream time series.

Different methodologies are proposed for stream data processing in mining stream time series such as Random Sampling method, Sliding windows method, Histograms method, Multiresolution methods etc., In this paper, we are using *sliding window model* to analyze stream data. The basic idea is that rather than running computations on the entire data, we can make decisions based only on recent data. More formally, at every time *t*, a new data element arrives. This element expires at $t + w$, where w is the window *size* or *length*. The sliding window model is useful for stocks or sensor networks, where only recent events may be important and reduces memory requirements because only a small window of data is used. In contrast, our work focuses on the detection of multiple static/dynamic patterns over multiple stream time series.

Contribution: In this paper, we present a new approach for representation of the stream time series data i.e., alternate data cluster or segment mean of stream time series data. The cost of computational time is saved by selecting alternate patterns from the stream time series and computing the mean cluster segment which takes logw steps. The pattern representation that is progressively computed is most suitable for regular and dynamic updation during processing of streams. This data representation works very well under all L_p-norms.

2 Related Works

Agrawal et al.[1] proposed whole sequence matching which finds sequences of the same length that are similar to a query sequence using Discrete Fourier Transform (DFT). Later, Floutsos et al. [2] extended this work to subsequence matching using efficient indexing method to locate 1-dimensional subsequences within a collection of sequences. In both the works Euclidean distance is used to measure the similarity among sequences. Guttman et al.[3] proposed a Dynamic Index structure for spatial searching.

Xian Lian et al.[4] proposed multiscale representations for fast pattern matching in stream time series using multiscale segment mean(MSM), which can be incrementally computed and perfectly adopted to the stream characteristics. Sethukkarasi et al.,[5] proposed a technique for similarity matching between static/dynamic patterns and stream time-series image data to perform effective retrieval of image data. Many techniques such as DFT, DWT[1], Piecewise Aggregate Approximation(PAA) and MSM[4] are used for dimensionality reduction problem.

3 Problem Definition and System Model

3.1 Problem Statement

Let us consider a set of stream time series data *D*, such that data stream, $D = (d_1, d_2, ..., d_i, ..., d_t)$, where each d_i is the real data arrival at a specific time period *i* and the index *t* is the current timestamp. *D* is used to compare with pattern set, $P = p_1, p_2, ..., p_m$, with the sliding window $W_i = (d_i, d_{i+1}, ..., d_i = w - 1)$, where *i*=1 ..., (*t* - *w*+1) and *w* is window size. In Multiscale representation sliding window W_i is similar to pattern p_i. if $dist(W_i, p_j) \leq \in$ where dist is in L_p norm function ($p \geq 1$) defined by the following equations (1) and (2).

$$L_p[\mathrm{X,Y}] = \sqrt[p]{\sum_{i=0}^{n-1} |X[i] - Y[i]|^p} \quad (1)$$

Where $p \geq 1$. If p=1, then it is called as Manhattan distance, If p=2, then it is called as Euclidean distance and when *p* is infinite, *L* norm can be defined as

$$L_\infty[\mathrm{X,Y}] = MAX_{i=0}^{n-1} \mid \mathrm{X[i]\text{-}Y[i]} \mid \quad (2)$$

The objective is to find similar patterns over a stream time series and reduce the processing cost and the dimensionality of time series W_i and pattern p_j. Here we assume a set of predefined time-series patterns, $P = p_1, p_2, ..., p_m$, where the length of each pattern p_j is equal to *w* and computational time complexity is reduced to O(log *w*). Whereas in MSM[4] the time complexity is O(n).

3.2 System Model

Alternate Data Cluster-Mean Pruning (ADCMP) Approximation

In this paper, we propose a new ADCMP approximation of time series data based on Multiscale Segment Mean (MSM) [4]. Figure 1 shows alternate data cluster mean representation of a time series data *D* of length w, where w=(2^l). We construct ADCMP approximation for the input time series data *D* by computing cluster mean from level 1 to level *l*. At each level *j*, there are (2^{j-1})) disjoint segments. Thus in the Figure 1 at different levels 3, 2 and 1 we can construct total of eight, four and two clusters respectively. The first cluster $S_{1,3}$ at level 3 is the mean of the first and third value in input stream time series data *D*. Similarly, the second cluster $S_{2,3}$ at level 3 is the mean of the fifth and seventh values in input stream time series data *D*. We can construct the clusters up to $S_{8,3}$ on level 3 as above.

At level 2, we can construct four clusters, each cluster is computed by the mean of two adjacent clusters on level 3. eg., $S_{1,2}$ = (($S_{1,3}$ + $S_{2,3}$)/2) and $S_{2,2}$ = (($S_{3,3}$S + $S_{4,3}$)/2) and so on. The ADCMP approximation for the input time series data *D* is represented by (S(W)= $D(W_1)$, $D(W_2)$, $D(W_l)$), where ($D(W_j)$) is cluster mean of *W* on level *j* for ($1 \leq j \leq 2^j$).

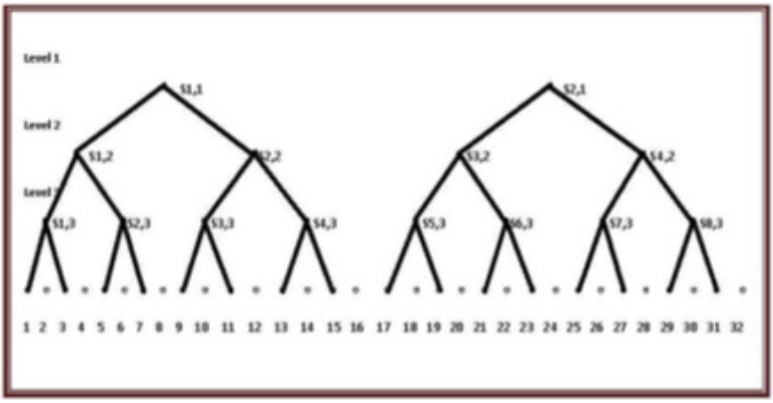

Fig. 1. Alternate data cluster mean representation

4 Algorithms

4.1 Alternate Data Cluster Mean Pruning (ADCMP)

ADCMP prunes patterns in P for Sliding Window W, by comparing the distance between approximation values of patterns and Sliding Window at each level. First, we retrieve resultant patterns in Grid Index *GI* which are within the user specified(∈) threshold distance from the Sliding Window *SW*. We use L_p norm distance with p=2 and a user defined threshold (∈), and the distance of pattern values in the Grid from the Approximation values of Sliding Window is within ($\in/2^l$), where l is the number of approximation levels. The pruned patterns are stored in Resultant Pattern *RP* and all the other patterns are filtered out of the grid storage. ADCMP algorithm gives the set of Resultant Patterns (*RP*) which are matched with Sliding Window (*SW*) of length ($wl= 2^l$) by taking input Pattern Set (*PS*) with user specified threshold similarity (∈) and a Grid Index (*GI*). This approach initially stores all the precomputed ADCMP approximation values(V) of patterns with IDs in a Grid Index(*GI*) structure. We start pruning from the highest level of approximation (l_{min}) i.e., from the root. A null set is initialized for Resultant Patterns (*RP*). Initially, we keep all the pattern values in the Grid.

4.2 Similarity Pattern-Matching (SPM)

This algorithm outputs the set of matching pairs by taking the inputs such as, stream time series *S*, a pattern set *P*, user specified threshold similarity (∈) and Grid Index *GI*. time series *S*, a pattern set *P*, user specified threshold similarity (∈) and Grid Index *GI*. SPM illustrates the query procedure to the pattern match between patterns and stream time series. The real distance between the candidate pattern and the stream series is found. If it is within (∈), then the similar pairs (W_i, *pt*) are retrieved.

Algorithm 1. Alternate Data Cluster Mean Pruning (ADCMP)

```
1: Let V be the approximation values of patterns in the pattern
set P_s that match with sliding window SW and through grid index
GI.
2: begin
3:      let i =l_min
4:      for each alternate data in the input stream time series
5:         Repeat
6:               Assign (RP=0; )
7:               Comparing V and SW at each level i
8:               Collect all Resultant Patterns (RP)
9:               if Dist (SW, P_s)≤ ∈/2(l+1-i))
10:                 Store all Resultant Patterns in RP
11:                 Replace GI by RP
12:                 Increment i by 1 to next level
13:              endif
14:        until ( (i≤ l)&(V! =0))
15:    endfor
16:    Return the patterns RP
17: end
```

Algorithm 2. Similarity Pattern-Matching (SPM)

```
1: Let W_i be sliding window with w recent values containing new
data item n_d
2: begin
3:       do
4:         Update the sliding window with recent w Values
5:         Calculate ADCMP approximations for new w.
           Prune all Reference patterns RP by ADCMP
6:          for each pattern pt in RP
7:             if dist (W_i, pt)≤ ∈)
8:                Output similar pairs (W_i, pt)
9:             endif
10:         endfor
11:      while no new data items found
12: end
```

5 Result Analysis

Experiments over the generated random data sets were performed. Such generated synthetic data is applied on the DWT, MSM and ADCMP algorithms. We observed that, the average CPU time required for existing DWT algorithm is 2.36 seconds, average CPU time required for MSM 1.904 seconds, whereas in our ADCMP algorithm, average CPU time required is 0.282 seconds. Our algorithm ADCMP is 80% more efficient than MSM and DWT algorithms as shown in Figure 2.

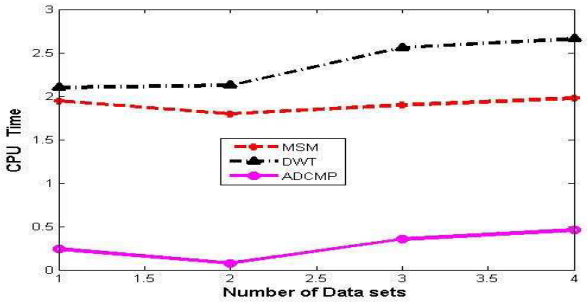

Fig. 2. Comparison of ADCMP MSM and DWT algorithm

6 Conclusion

In this paper, we proposed an algorithm ADCMP, which performs high-speed pattern match on stream time series data, where the data is pruned with computational cost of O(log *w*), which results in better performance than Multiscale Segment Mean[4] and DWT[1]. This approach further can be applied on multidimensional data like streaming video data and multimedia data.

References

1. Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient Similarity Search in Sequence Databases. In: Proc. Fourth Intl Conf. Foundations of Data Organization and Algorithms, FODO (1993)
2. Faloutsos, C., Ranganathan, M., Manolopoulos, Y.: Fast Subsequence Matching in Time-Series Databases. In: Proc. ACM SIGMOD (1994)
3. Guttman, A.: R-Trees: A Dynamic Index Structure for Spatial Searching. In: Proc. ACM SIGMOD (1984)
4. Lian, X., Chen, L., Yu, J.X., Han, J., Ma, J.: Multiscale Representations for Fast Pattern Matching in Stream Time Series. The IEEE TKDE 21(4), 568–581 (2009)
5. Sethukkarasi, R., Rajalakshmi, D., Kannan, A.: Efficient and Fast Pattern Matching in Stream Time Series Image Data. In: Proc. 1st ICIIC (2010)