An Efficient Method for Discovering Motifs in Streaming Time Series Data

Cao Duy Truong and Duong Tuan Anh

Abstract. The discovery of repeated subsequences, *time series motifs*, is a problem which has great utility for several higher-level data mining tasks, including classification, clustering, forecasting and rule discovery. In recent years there has been significant research effort spent on efficiently discovering these motifs in static time series data. However, for many applications, the streaming nature of time series demands a new kind of methods for discovery of time series motifs. In this paper, we develop a new method for motif discovery in streaming time series. In this method we use significant extreme points to determine motif candidates and then cluster motif candidates by BIRCH algorithm. The method is very effective not only for large time series data but also for streaming environment since it needs only one-pass of scan through the whole data.

1 Introduction

A time series is a sequence of real numbers measured at equal intervals. Time series data arise in so many applications of various areas ranging from science, engineering, business, finance, economic, medicine to government. There are two kinds of time series data: time series in static environment and time series in high speed data stream environment. In streaming time series database, the database changes continuously as new data points arrive continuously. Examples of the streaming time series applications are online stock analysis, computer network monitoring, network traffic management, earthquake prediction. Streaming time series have their own characteristics, compared to static time series: (1) Data are frequently updated in stream time series, thus, previous approaches applied to static time series may not work in the scenario. (2) Owing to the frequent updates, it is impossible to store all the data in memory, thus, efficient and one-pass algorithms are very important to achieve a

Cao Duy Truong · Duong Tuan Anh
Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology
e-mail: caoduytruong@hcmunre.edu.vn, dtanh@cse.hcmut.edu.vn

DOI: 10.1007/978-3-319-02741-8_20, © Springer International Publishing Switzerland 2014

real time response. Data mining on streaming environment is recently considered as one of the top ten challenging problems in the data mining research field [15].

Time series motifs are frequently occurring but previously unknown subsequences of a longer time series. One major challenge in motif discovery for time series data is the large volume of the time series data. Since the first formal definition of time series motif given by Lin et al. in 2002 [6], several algorithms have been proposed to tackle time series motif discovery. The first algorithm ([6]) defines the problem of motif discovery in time series regarding two parameters: (1) the motif length *m* and (2) a user-defined range parameter *r*. Some of these algorithms tackle the motif discovery by first applying some dimensionality reduction transformations such as PAA, PLA and some discretization techniques such as SAX, iSAX, ([1], [2], [5], [6], [14], [16]). Some of the algorithms aim to discovering motif with different lengths or discovering motif with the suitable length determined automatically ([11], [12]). However, so far surprisingly there have been very few research works on time series motif discovery in streaming environment.

In 2010, a motif discovery method for streaming time series was proposed by Mueen and Keogh, which was considered as “the first practical algorithm for finding and maintaining time series motifs on fast moving streams” [8]. However, in this work, the authors use the new *nearest-neighbor* motif definition by which time series motif is defined as a closest pair of subsequences in time series data. This definition does not take into account the frequency of the subsequences, therefore, the motif discovery algorithm proposed in [8] is not convenient to be used directly in practical applications.

In this paper, we propose an efficient method for motif discovery in streaming time series which adopts the first formal motif definition given in 2002 [6]. Our proposed method needs only one single pass over the whole data, and can update the motif results efficiently whenever there have been some new data points arriving. Our method works directly on the raw data without using any transformation for dimensionality reduction or discretization. The instances of a motif discovered by our method may be of different lengths and user does not have to predefine the length of the motif. The proposed method requires fewer user-predefined parameters and these parameters are easy to be determined through experiments. The proposed method is also not sensitive to the changes of these parameters. Our method uses significant extreme points to determine motif candidates and then clusters motif candidates to find the most significant motif by using BIRCH algorithm.

The rest of the paper is organized as follows. In Section 2 we explain briefly some basic backgrounds on time series motif and our previous work on time series motif discovery in static time series. Section 3 introduces the proposed method. Section 4 reports experimental results on the proposed method. Section 5 gives some conclusions and remarks for future work.

2 Background

In this section we introduce some useful definitions and the EP-BIRCH algorithm that can discover motif in static time series.

2.1 Time Series Motif

Definition 1. *Time Series*: A time series $T = t_1, , t_N$ is an ordered set of N real-values measured at equal intervals.

Definition 2. *Similarity distance*: $D(s_1, s_2)$ is a positive value used to measure differences between two time series s_1, and s_2, relies on measure methods. If $D(s_1, s_2) < r$, where r is a real number (called range), then $s1$ is similar to $s2$.

Definition 3. *Subsequence*: Given a time series T of length N, a subsequence C of T is a sampling of length $n < N$ of contiguous positions from T, that is, $C = t_p, ..., t_{p+n-1}$ for $1 < p < N - n + 1$.

Definition 4. *Time series motif*: Given a time series T, a subsequence C is called the most significant motif (or *1-motif*) of T, if it has the highest count of the subsequences that are similar to it. All the subsequences that are similar to the motif are called instances of the motif.

This definition is also the first formal definition of time series motif given by Lin et al. in 2002 [6].

Definition 5. The *motif count* (or *frequency*) of a motif M is the total number of instances that M has in time series T.

Time series motifs are typically sorted according to their motif count. The *K-motif* is the motif ranked at *K-th* position regarding number of instances.

2.2 The EP-BIRCH Algorithm

Finding Significant Extreme Points

To extract a temporally ordered sequence of motif candidates, significant extreme points of a time series have to be found. The definition of significant extreme points, given by Pratt and Fink, 2002 [10] is as follows.

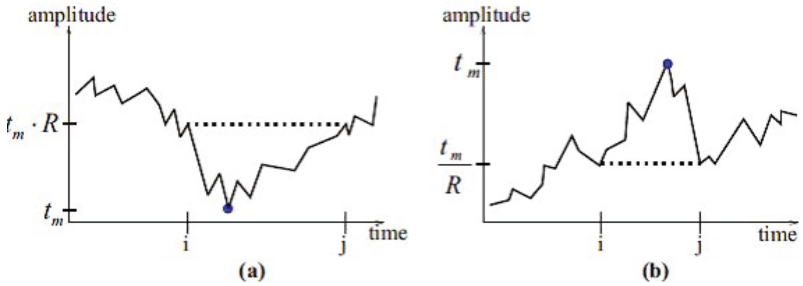

Fig. 1 Illustration of Significant Extreme Points: (a) Minimum, (b) Maximum

Definition 6. *Significant Extreme Points*: A univariate time series $T = t_1, ..., t_N$ has a significant minimum at position m with $1 < m < N$, if $(t_i, ..., t_j)$ with $1 \leq i < j \leq N$ in T exists, such that t_m is the minimum of all points of this subsequence and $t_i \geq R \times t_m$, $t_j \geq R \times t_m$ with user-defined $R \geq 1$.

Similarly, a significant maximum is existent at position m with $1 < m < N$, if a subsequence $(t_i,...,t_j)$ with $1 \leq i < j \leq N$ in T exists, such that t_m is the maximum of all points of this subsequence and $t_i \leq t_m/R, t_j \leq t_m/R$ with user-defined $R \geq 1$.

Notice that in the above definition, the parameter R is called *compression rate* which is greater than one and an increase of R leads to selection of fewer significant extreme points. Fig. 1. illustrates the definition of significant minima (a) and maxima (b). Given a time series T, starting at the beginning of the time series, all significant minima and maxima of the time series are computed by using the algorithm given in [10].

The significant extreme points can be the starting point or ending point of a motif instances. Basing on the extracted significant points we can extract the motif candidates from a time series and then cluster them using BIRCH algorithm.

BIRCH Clustering

BIRCH is designed for clustering a large amount of numerical data by integration of hierarchical clustering at the initial stage and other clustering methods, such as iterative partitioning at the later stage [17]. It introduces two main concepts, *clustering feature and clustering feature tree (CF tree)*, which are used to summarize cluster representations. These structures help the clustering method achieve good speed and scalability in large databases. BIRCH is also effective for *incremental* and *dynamic clustering* of incoming objects.

Given N d-dimensional points or objects $\overrightarrow{x_i}$ in a cluster, we can define the centroid $\overrightarrow{x_0}$, the radius R, and the diameter D of the cluster as follows:

$$\overrightarrow{x_0} = \frac{\sum_{i=1}^{N} \overrightarrow{x_i}}{N} \tag{1}$$

$$D = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\overrightarrow{x_i} - \overrightarrow{x_j})^2}{N(N-1)}} \tag{2}$$

$$R = \sqrt{\frac{\sum_{i=1}^{N} (\overrightarrow{x_i} - \overrightarrow{x_0})^2}{N}} \tag{3}$$

where R is the average distance from member objects to the centroid, and D is the average pairwise distance within a cluster. Both R and D reflect the tightness of the cluster around the centroid. A clustering feature *(CF)* is a triplet summarizing information about clusters of objects. Given N d-dimensional points or objects in a subcluster, then the CF of the cluster is defined as

$$CF = (N, \overrightarrow{LS}, SS) \tag{4}$$

where N is the number of points in the subcluster, $\overrightarrow{LS}$ is the linear sum on N points and SS is the square sum of data points.

$$\overrightarrow{LS} = \sum_{i=1}^{N} \overrightarrow{x_i} \tag{5}$$

$$SS = \sum_{i=1}^{N} \overrightarrow{x_i}^2 \tag{6}$$

A clustering feature is essentially a summary of the statistics for the given subcluster: the zero-th, first, and second moments of the subcluster from a statistical point of view. Clustering features are additive. For example, suppose that we have two disjoint clusters, C_1 and C_2, having the clustering features, CF_1 and CF_2, respectively. The clustering feature for the cluster that is formed by merging C_1 and C_2 is simply $CF_1 + CF_2$. Clustering features are sufficient for calculating all of the measurements that are needed for making clustering decisions in BIRCH.

A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering. By definition, a nonterminal node in the tree has descendents or "children". The nonleaf nodes store sums of the CFs of their children, and thus summarize clustering information about their children. Each entry in a leaf node is not a single data objects but a subcluster. A CF tree has two parameters: branching factor (B for nonleaf node and L for leaf node) and threshold T. The branching factor specifies the maximum number of children in each nonleaf or leaf node. The threshold parameter specifies the maximum diameter of the subcluster stored at the leaf nodes of the tree. The two parameters influence the size of the resulting tree.

BIRCH applies a multiphase clustering technique: a single scan of the data set yields a basic good clustering, and one or more additional scans can (optionally) be used to further improve the quality. The BIRCH algorithm consists of four phases as follows.

Phase 1: *(Building CF tree)* BIRCH scans the database to build an initial in-memory CF tree, which can be view as a multilevel compression of the data that tries to preserve the inherent clustering structure of the data.

Phase 2: *[optional]* (Condense data) Condense into desirable range by building a smaller CF tree.

Phase 3: *(Global Clustering)* BIRCH applies a selected clustering algorithm to cluster the leaf nodes of the CF tree. The selected algorithm is adapted to work with a set of subclusters, rather than to work with a set of data points.

Phase 4: *[optional]* Cluster refining

After the CF tree is built, any clustering algorithm, such as a typical partitioning algorithm, can be used in Phase 3 with the CF tree built in the previous phase. Phase 4 uses the centroids of the clusters produced by Phase 3 as seeds and redistributes the data points to its closest seed to obtain a set of new clusters.

EP-BIRCH Algorithm

The EP-BIRCH (Extreme points and BIRCH clustering) method, introduced in our previous paper [13], is an improvement of the EP-C algorithm proposed by Gruber et al. [3] for time series motif discovery. The EP-C algorithm uses hierarchical agglomerative clustering (HAC) algorithm for clustering which is not suitable to large scale time series datasets. In our EP-BIRCH method, we use BIRCH algorithm to cluster motif candidates rather than using HAC algorithm. BIRCH is especially suitable for clustering very large time series datasets. Besides, in the EP-C algorithm, each motif candidate is determined by three contiguous extreme points, but in our proposed method, motif candidate is determined by n contiguous extreme points where n is selected by user.

EP-BIRCH consists of the following steps:

Step 1: We extract all significant extreme point of the time series T. The result of this step is a sequence of extreme points $EP = (ep_1, ..., ep_l)$

Step 2: We compute all the motif candidates iteratively. A motif candidate $MC_i(T), i = 1, ..., l - n + 1$ is the subsequence of T that is bounded by the n extreme points ep_i and ep_{i+n-1}. Motif candidates are the subsequences that may have different lengths.

Step 3: Motif candidates are the subsequences that may have different lengths. To enable the computation of distances between them, we can bring them to the same length using *homothetic transformation*. The same length here is the average length of all motif candidates extracted in Step 2.

Step 4: We build the CF tree with parameters B and T. We insert to the CF tree all the motif candidates found in Step 3. We apply k-Means as Phase 3 of BIRCH to cluster the leaf nodes of the CF tree where k is equal to the number of the leaf nodes in the CF tree.

Step 5: Finally we find the subcluster in the CF tree with the largest number of objects. The *1-motif* will be represented by that cluster.

In the Step 3, to improve the effectiveness of our proposed method, we apply *homothety* for transforming the motif candidates with different lengths to those of the same length rather than spline interpolation as suggested in [3]. Spline interpolation is not only complicated in computation, but also can modify undesirably the shapes of the motif candidates. Homothety is a simpler and more effective technique which also can transform the subsequences with different lengths to those of the same length. Due to the limit of space, we can not explain the use of homothetic transformation here, interested readers can refer to [13] for more details.

3 The Proposed Method for Motif Discovery in Streaming Time Series Data

3.1 From EP-BIRCH to EP-BIRCH-STS

Our method for motif discovery in streaming time series is developed from our previous work, the EP-BIRCH algorithm for motif discovery in static time series [13].

We call the new method EP-BIRCH-STS (Extreme Points and BIRCH clustering for discovering motif in Streaming Time Series). Our method works with the features extracted from the whole time series from the beginning data point to the newest incoming data point. In our method, we can view the sequence of extracted extreme points $EP = (ep_1, ..., ep_l)$ as a simple data structure to keep all the features of the time series. In the proposed method for motif discovery in streaming time series, EP-BIRCH-STS, we require the following parameters.

- R: compression rate for computing the significant extreme points.
- min_Length and max_Length: the lower bound and upper bound for the length of the discovered motif.
- B, T: branching factor and threshold of CF tree.

EP-BIRCH-STS is mainly based on EP-BIRCH with some following modifications in order to adapt it in the streaming environment

1. In EP-BIRCH algorithm, when finding the significant extreme points, the algorithm uses two subroutines: FIND-MINIMUM(i) for finding the significant minimum starting from the $i-th$ point in the time series and FIND-MAXIMUM(i) for finding the significant minimum starting from the $i-th$ point in the time series. We can make these two subroutines incremental to accommodate the new incoming data points. So, we can make the task of extracting significant extreme points incremental.
2. We apply a *deferred update policy* that is described as follows. We delay the update of the sequence of extreme points $EP = (ep_1, ..., ep_l)$ until we identify one new significant extreme point from new coming data points and this event actually yields a new motif candidate. At that moment, we activate the motif discovery phase in order to get a new possible motif for the current time series data. When we have the new motif candidate, we apply the homothety to convert it to a suitable length and then insert it to the CF-tree. Due to the incremental nature of BIRCH clustering algorithm, the clustering step in EP-BIRCH-STS can work very efficiently to produce a new possible motif result whenever there is an incoming motif candidate.

3.2 Post Processing

In the paper [4], Keogh and Lin, 2005 pointed that subsequence clustering in streaming time series is meaningless when we use a sliding window to extract subsequences. The task is meaningless since there are several trivial matches (Definition 7) during clustering.

Definition 7. *Trivial Match*: Given a subsequence C beginning at position p, a matching subsequence M beginning at q, and a distance R, we say that M is a trivial match to C of order R, if either $p = q$ or there does not exist a subsequence M beginning at q such that $D(C,M) > R$, and either $q < q < p$ or $p < q < q$.

The subsequence extraction in our EP-BIRCH-STS method which based on significant extreme points does not use sliding window and therefore its clustering step does not have the problem mentioned in [4]. However to ensure there exists no trivial matches, in EP-BIRCH-STS, we perform a post processing step which excludes any trivial matches.

In our EP-BIRCH-STS method, trivial matches may arise when the compression *R* is too small. When *R* is too small, the extreme points will be not enough significant and this make the motif candidates in a subcluster to be overlapped with one another. When *R* is large enough, trivial matches can not arise since each extreme point starts a new change in the time series. Fig. 2. illustrates the possibility of trivial matches among motif candidates when R is too small.

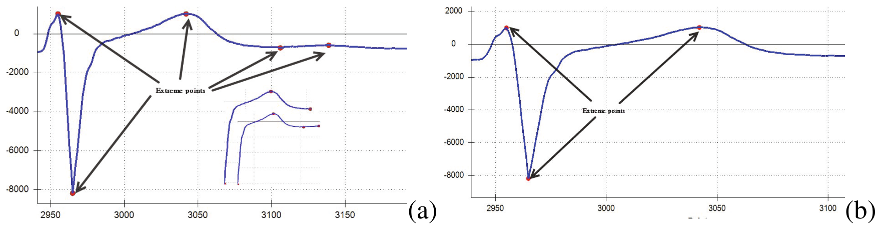

Fig. 2 Trivial matches due to the compression ratio *R*. (a) With too small *R*, motif candidates may start with the same position, but belong to the same subcluster. (b) With large *R*, there is no trivial match.

To exclude possible trivial matches in the very rare cases, after obtaining the subclusters by using BIRCH algorithm, we examine in all the subclusters to exclude any instances which have some overlap in one another. This task takes very low computation cost. Then we rank the resultant subclusters according to the number of instances in them in order to determine the top subcluster as *1-motif* and the *K-th* subcluster as *K-motif*.

4 Experimental Evaluation

In this experiment, we compare our method, EP-BIRCH-STS to the modified version of Online-MK method, the first motif discovery method for streaming time series ([8]). We implemented the two methods with Microsoft Visual C# and conducted the experiment on a Core i7, Ram 4GB PC.

First, to verify the correctness of EP-BIRCH-STS, we experiment the method on the datasets which were used in four previous works ([5], [7], [8], [9]). The test datasets consist of.

1. ECG (electrocardiogram) dataset.[1]
2. Insect behavior dataset.[2]
3. World cup dataset.[3]

[1] `http://www.cs.ucr.edu/~eamonn/iSAX/koski_ecg.dat`
[2] `http://www.cs.ucr.edu/~mueen/txt/insect15.txt`
[3] `http://www.cs.ucr.edu/~mueen/txt/wc_index.txt`

Experimental results show that when the appropriate values are selected for the parameters, the motifs found by EP-BIRCH-STS are exactly the same as those discovered in [5] and [9]. Fig. 3. reports the motifs found by our EP-BIRCH-STS method in comparison to those discovered in [5] and [9].

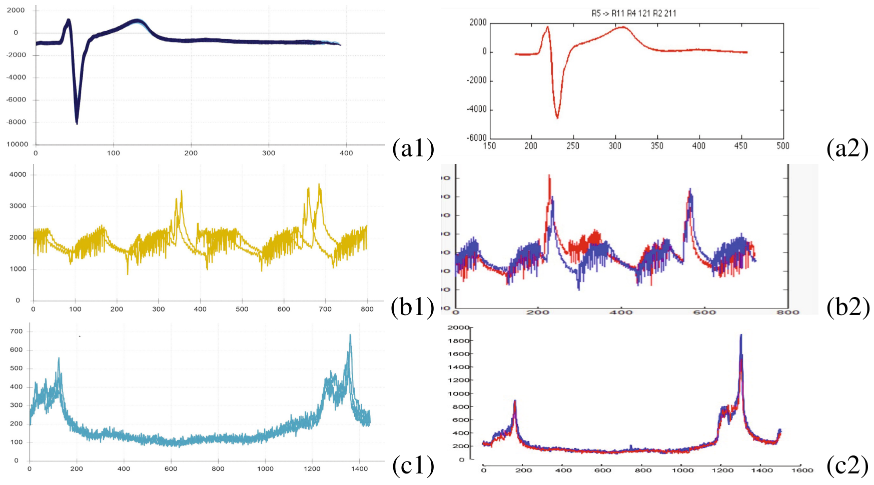

Fig. 3 Motifs discovered by the proposed method compared with those found by some previous methods. (a1), (a2) motif found in *ECG dataset* by EP-BIRCH-STS and [5]. (b1), (b2) motif found in *insect behavior dataset* by EP-BIRCH-STS and [5]. (c1), (c2) motif found in *world cup dataset* by EP-BIRCH-STS and [9].

To verify the effectiveness of our EP-BIRCH-STS in streaming time series, we conduct an experiment which accepts the above-mentioned datasets in stream manner after a fixed time period. Fig. 4. shows the 1-motifs discovered from the ECG dataset with lengths from 200 to 300 at different time points.

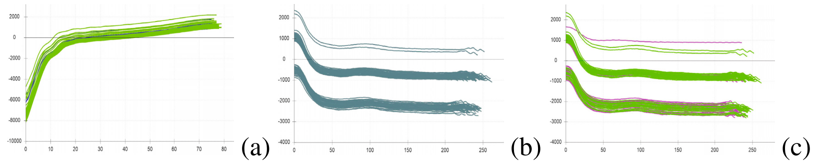

Fig. 4 Motif in *ECG dataset* at different time points. (a) At time point 71996 with 80 instances; (b) at time point 79701 with 81 instances; (c) at time point 144404 with 110 instances.

Next, we compare the performance of EP-BIRCH-STS to the modified version of Online-MK method described in [8]. We use the same datasets that were used in the experiment reported in [8]. They are EEG trace, EOG trace, insect behavior trace and a synthetic random walk (RW). Since Online-MK method aims to find the most closest pair in a streaming time series while our method aims to find the 1-motif in the sense of its first formal definition (Definition 4 in Section 2), we have to modify the Online-MK so that it can discover 1-motif (in its first formal definition). We have to add to the Online-MK the following modifications:

- We specify the distance range r for the 1-motif discovery algorithm.
- We maintain a linked list, called neighbor list ($N-list$) for each subsequence as in Online-MK. But each N-list of a subsequence keeps only the subsequences whose distances to the subsequence under question are less than or equal to the range r.
- We select the subsequence which N-list contains the highest count of its neighbors as the 1-motif result.

We named the modified version of Online-MK as Modified-Online-MK. Since the Modified Online-MK finds 1-motif in the data segment within one sliding window while our EP-BIRCH-STS method discovers 1-motif in the whole time series, we carry out the experiment only on the data segments which are of the same length as the sliding window to ensure the comparison to be fair. In the experiment we use the compression rate $R = 1$ in order to extract the largest number of extreme points in the time series, i.e., even any data point which exhibits a small change in the time series can be considered as extreme point. Besides, in EP-BIRCH-STS, we apply homothetic transformation to convert all the motif candidates to the same length of the motif we desire to discover in Modified-Online-MK.

Table 1 Efficiency ratio of EP-BIRCH-STS vs. Modified-Online-MK

Window length	**1000**	**2000**	**4000**	**8000**	**10000**	**20000**
EEG	1.64%	0.30%	0.14%	0.07%	0.06%	0.03%
EOG	3.09%	0.83%	0.37%	0.25%	0.25%	0.25%
Insect	4.14%	1.23%	0.51%	0.26%	0.23%	0.14%
RW	2.33%	0.78%	0.35%	0.19%	0.15%	0.11%

To compare the efficiency of the two methods, we compute the efficiency ratio of EP-BIRCH-STS versus Modified-Online-MK which is defined as follows:

Efficiency ratio = the number of Euclidean distance function calls in EP-BIRCH-STS * 100% / the number of Euclidean distance function calls in Modified-Online-MK.

Table 1 show the efficiency ratio of EP-BIRCH-STS vs. Modified-Online-MK on different datasets and with different data lengths. Experimental results show that in EP-BIRCH-STS the number of Euclidean distance function calls just is about 0.74% of that in Modified-Online-MK. This remarkable performance of EP-BIRCH-STS is due to the fact that EP-BIRCH-STS uses the significant extreme points to extract motif candidates and this brings out a much smaller number of motif candidates in comparison to those extracted by Modified-Online-MK. Furthermore, by using CF-tree to support in clustering, the cost of subsequence matching in EP-BIRCH-STS is reduced remarkably since each motif candidate has to be matched with a smaller number of the nodes in CF-tree.

Table 2 show the CPU time (in seconds) of motif discovery in the two methods EP-BIRCH-STS and Modified-Online-MK respectively on different datasets and with different motif lengths. We can see that in all cases, the CPU time of motif discovery in EP-BIRCH-STS is much lower than that of Modified-Online-MK.

Table 2 Run time of EP-BIRCH-STS/run time of Modified-Online-MK on different datasets and with different motif lengths

Motif length	**64**	**128**	**256**	**396**	**512**
EEG	0.38s/58.82s	0.47s/120.41s	0.63s/264.23s	0.75s/430.94s	0.68s/570.58s
EOG	0.75s/11.77s	0.93s/102.95s	1.35s/223.25s	1.63s/380.93s	1.58s/519.39s
Insect	0.84s/76.34s	1.16s/133.55s	1.50s/227.95s	1.63s/351.71s	1.54s/447.03s
RW	0.78s/76.92s	0.98s/133.17s	1.24s/229.64s	1.48s/350.67s	1.57s/482.21s

5 Conclusions

We haved introduced a new method for discovering motifs in streaming time series. This method, called EP-BIRCH-STS, is based on extracting significant extreme points and clustering the motif candidates by using BIRCH algorithm. This method needs only one-pass scan through the whole data and can discover 1-motif with the length in the range of *min_Length* to *max_Length*. The discovered motif some-whatdepends on the values of the compression rate R specified in the step of identifying the significant extreme points and the threshold T in the BIRCH clustering algorithm.

As for future work, we plan to find some technique to determine the two parameters R,and T automatically for any given time series dataset.

References

1. Castro, N., Azevedo, P.J.: Multiresolution Motif Discovery in Time Series. In: Proc. of the SIAM Int. Conf. on Data Mining, SDM 2010, Columbus, Ohio, USA, April 29-May 1 (2010)
2. Chiu, B., Keogh, E., Lonardi, S.: Probabilistic Discovery of Time Series Motifs. In: Proc. of 9th Int. Conf. on Knowledge Discovery and Data Mining (KDD 2003), pp. 493–498 (2003)
3. Gruber, C., Coduro, M., Sick, B.: Signature verification with dynamic RBF network and time series motifs. In: Proc. of 10th International Workshop on Frontiers in Hand Writing Recognition (2006)
4. Keogh, E., Lin, J.: Clustering of Time-Series Subsequences is Meaningless. Implications for previous and future research. Knowl. Inf. Syst. 8(2), 154–177 (2005)
5. Li, Y., Lin, J., Oates, T.: Visualizing Variable-Length Time Series Motifs. In: SDM 2012, pp. 895–906 (2012)

6. Lin, J., Keogh, E., Patel, P., Lonardi, S.: Finding Motifs in Time Series. In: Proc. of the 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002)
7. Mueen, A., Keogh, E., Zhu, Q., Cash, S., Westover, B.: Exact Discovery of Time Series Motif. In: Proc. of 2009 SIAM International Conference on Data Mining, pp. 1–12 (2009)
8. Mueen, A., Keogh, E.: Online Discovery and Maintenance of Time Series Motif. In: Proc. of ACM SIGKDD 2010, pp. 1089–1098 (2010)
9. Mueen, A., Keogh, E., Zhu, Q., Cash, S., Bigdely-Shamlo, N.: Finding Time Series Motifs in Disk-Resident Data. In: Proc. of IEEE International Conference on Data Mining, ICDM, pp. 367–376 (2009)
10. Pratt, K.B., Fink, E.: Search for patterns in compressed time series. International Journal of Image and Graphics 2(1), 89–106 (2002)
11. Tanaka, Y., Iwamoto, K., Uehara, K.: Discovery of Time Series Motif from Multi-Dimensional Data based on MDL Principle. Machine Learning 58(2-3), 269–300 (2005)
12. Tang, H., Liao, S.: Discovering Original Motifs with Different Lengths from Time Series. Knowledge-based System 21(7), 666–671 (2008)
13. Truong, C.D., Anh, D.T.: An Efficient Method for Discovering Motifs in Large Time Series. In: Selamat, A., Nguyen, N.T., Haron, H. (eds.) ACIIDS 2013, Part I. LNCS (LNAI), vol. 7802, pp. 135–145. Springer, Heidelberg (2013)
14. Xi, X., Keogh, E., Li, W., Mafra-neto, A.: Finding Motifs in a Database of Shapes. In: SDM 2007. LNCS, vol. 4721, pp. 249–260. Springer, Heidelberg (2007)
15. Yang, Q., Wu, X.: 10 Challenging Problems in Data Mining Research. Intl. Jrnl. of Information Technology & Decision Making 5(4), 597–604 (2006)
16. Yankov, D., Keogh, E., Medina, J., Chiu, B., Zordan, V.: Detecting Time Series Motifs under Uniform Scaling. In: Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2007), pp. 844–853 (2007)
17. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: An efficient data clustering method for very large databases. SIGMOD Rec. 25(2), 103–114 (1996)