Some Efficient Segmentation-Based Techniques to Improve Time Series Discord Discovery

Huynh Thi Thu Thuy(✉), Duong Tuan Anh, and Vo Thi Ngoc Chau

Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam
huynhthithuthuy@tdt.edu.vn, {dtanh,chauvtn}@cse.hcmut.edu.vn

Abstract. Time series discord has proved to be a useful concept for time series anomaly detection. To search for discords, various algorithms have been developed. HOT SAX has been considered as a well-known and effective algorithm in time series discord discovery. However this algorithm still has some weaknesses. First, users of HOT SAX are required to choose suitable values for the discord length, word-length and/or alphabet-size, which are unknown. Second, HOT SAX still suffers from high computation cost. In this paper, we propose some novel techniques to improve HOT SAX algorithm. These techniques consist of (i) using some time series segmentation methods to estimate the two important parameters: discord length and word length and (ii) speeding up the discord discovery process by a new way of shifting the sliding window. Extensive experiments have demonstrated that the proposed approach can not only facilitate users in setting the parameters, but also improve the discord discovery in terms of accuracy and computational efficiency.

Keywords: Time series · Discord discovery · HOT SAX · Segmentation

1 Introduction

The problem of detecting unusual (abnormal, novel, deviant, anomalous, *discord*) subsequences in a time series has recently attracted much attention. Time series anomaly detection brings out the abnormal patterns embedded in a time series. Areas that explore such time series anomalies are, for example, fault diagnostics, intrusion detection, fraud detection, auditing and data cleansing. Anomaly detection is a challenging topic, mainly because we need to obtain the lengths of anomaly patterns before detecting them.

Some popular algorithms for time series anomaly detection include window-based methods such as brute-force and HOT SAX by Keogh et al. (2005) [6] and WAT by Bu et al. (2007) [1]; a method based on segmentation and Finite State Automata by Salvador and Chan (2005) [15]; a method based on neural-network by Oliveira et al. (2004) [13]; a method based on time series segmentation and anomaly scores by Leng et al. (2008) [9]; a method based on PAA bit representation and clustering by Li et al. (2013) [10]; and a method which applies cluster-based outlier detection by Kha and Anh (2015) [8]. Among these above-mentioned algorithms for time series discord discovery, HOT SAX has been considered as the most popular algorithm. HOT SAX is an unsupervised

P.C. Vinh et al. (Eds.): ICTCC 2016, LNICST 168, pp. 179–188, 2016.
DOI: 10.1007/978-3-319-46909-6_17

method of anomaly detection and has been applied in several real applications. However, this algorithm still has some weaknesses. First, users of HOT SAX are required to choose suitable values for the discord length, word-length and/or alphabet-size, which are not intuitive. Second, HOT SAX still suffers from high computation cost and cannot satisfy the requirement of real applications with large datasets.

Time series segmentation can be considered as a preprocessing step and core task for variety of data mining tasks [12]. Segmentation focuses on dividing the time series into appropriate, internally homogeneous segments so that the structure of time series, through pattern discovery in the behavior of the observed variable, could be revealed. Segmentation is not a trivial problem. Common segmentation methods include Piecewise Linear Approximation (PLA) by Keogh et al. (2002) [5], major extreme points by Fink and Pratt (2002) [14], and Perceptually Important Points (PIP) proposed by Fu et al. (2006) [3]. Segmentation has been used in some time series data mining tasks, which can be listed as follows. Salvador and Chan (2005) [15] applied segmentation and Finite State Automata to detect anomaly in time series. Gruber et al. (2006) [4] applied major extreme points in identifying candidate patterns for time series motif discovery. Leng et al. (2008) [9] used segmentation and anomaly scores for anomaly detection in time series with dynamic time warping distance. Dani et al. (2015) used segmentation and local means and standard deviations for time series anomaly detection [2]. Through these previous works, it is obvious that there is a strong relationship between time series segments brought out by a segmentation method and the meaningful patterns for motif/anomaly discovery in time series. Motivated by this direction, in this work we attempt to apply some segmentation techniques in improving HOT SAX, especially in estimating two important parameters n and w.

In this work, we propose some novel techniques to improve HOT SAX algorithm. These techniques consist of (i) using two time series segmentation methods (PLA and major-extreme points) to estimate the two important parameters in HOT SAX: discord length n and word length w; and (ii) speeding up the discord discovery process by shifting the sliding window one PAA frame at a time. Extensive experiments on several datasets have demonstrated that the proposed approach can not only facilitate users in setting the parameters but also improve discord discovery in terms of accuracy and computational efficiency.

2 Background and Related Works

In this section, we introduce some background on time series discords, HOT SAX algorithm and some basic ideas on time series segmentation.

2.1 Time Series Discords

According to Keogh et al., 2005 [6] a time series discord is a subsequence that is very different from its closest matching subsequence. However, in general, the best matches of a given subsequence tend to be very close to the subsequence under consideration.

Such matches are called *trivial matches* and are not useful. When finding discords, we should exclude trivial matches and keep only *non-self matches* as defined as follows.

Definition 1 (*Non-self match*): Given a time series T containing a subsequence C of length n beginning at position p and a matching subsequence M beginning at the position q, we say that M is a non-self match to C if $|p - q| \geq n$.

Definition 2 (*Time series discord*): Given a time series T, the subsequense D in T is called the most significant discord in T if the distance to its nearest non-self match is largest.

The naïve method for finding discords (called Brute Force Discord Discovery algorithm - BFDD) is also given in [6]. The algorithm is implemented by a two layer nested loop. The outer loop considers each possible candidate subsequence and the inner loop is a linear scan to find the nearest non-self match of the candidate. The subsequence that has the largest distance to its nearest non-self match is the discord. This method is simple, easy to implement, and can produce the exact solution. The main problem is that it has $O(m^2)$ time complexity where m is the length of the time series.

2.2 HOT SAX Algorithm

The BFDD algorithm is mainly designed for raw time series. By applying time series dimensionality reduction techniques, discord can be detected also based on approximate representation. HOT SAX algorithm, proposed by Keogh et al. [6], is the heuristic method which applies Symbolic Aggregate Approximation (SAX) representation [13] into a discord discovery algorithm. To symbolize a time series by SAX representation, first, HOT SAX has to reduce its dimensionality by applying Piecewise Aggregate Approximation (PAA) transform [11].

Having converted a time series into the PAA representation, HOT SAX applies a further transformation to obtain a discrete SAX representation. Notice that SAX representation requires that the data should meet Gaussian distribution. To find discords of length n in a time series T, HOT SAX first shifts a sliding window of length n across time series T, extracting subsequences, encoding them into charactering strings (called *SAX words*) and placing them in a table where the index refers back to the original subsequence. Once having this ordered list of SAX words, HOT SAX can store them in a tree structure (called *augmented trie*) where the leaf nodes keep a linked list of all word occurrences that map there. HOT SAX relies on three important parameters: the length of discords n, the cardinality of the SAX alphabet a, and the SAX word size w.

To realize early exit of the for loops, HOT SAX applies the two following heuristics. (a) In the outer loop, the subsequence with larger distance to its nearest neighbor has a priority to be selected for comparison. This kind of subsequences corresponds to the entries in the leaf nodes of the augmented trie which have the small word count. (b) In the inner loop, the subsequence with smaller distance to the current candidate has a priority to be compared. This kind of subsequences corresponds to the entries in the same leaf node with the current candidate.

2.3 Time Series Segmentation

In this work, we try to apply some segmentation techniques in improving HOT SAX algorithm, especially in estimating two important parameters: discord length n and word length w. Among many popular time series segmentation methods, we select to use the two most well-known methods: PLA and a method which based on major extreme points.

Identifying Major Extreme Points. Pratt and Fink [14] proposed a method for compressing time series which is based on the concept of major extreme points. But in this work, we apply the concepts of major extreme points for time series segmentation rather than for time series compression.

Major extreme points in a time series contain important features of the time series. The definition of major extreme points is given as follows.

Definition 3. *Major extreme points:* A point t_k of a time series $T = t_1,\ldots, t_m$ is a *major minimum* if there are indices i and j, where $i < k < j$, such that t_k is the minimum among $t_i,\ldots, t_j$ and $t_i/t_k \geq R$, and $t_j/t_k \geq R$.

Intuitively, t_k is a minimum value of some segment $t_i,\ldots, t_j$ and the endpoint values of this segment are much larger than t_k. Similarly, a point t_k is an *major maximum* if there are indices i and j, where $i < k < j$, such that t_k is the maximum among $t_i,\ldots, t_j$ and $t_k/t_i \geq R$, and $t_k/t_j \geq R$.

Figure 1 illustrates the definition of major minima and maxima.

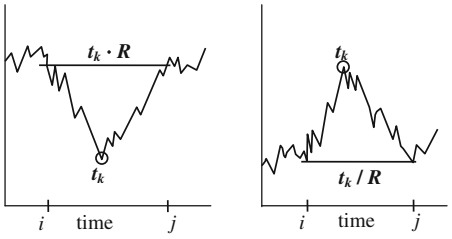

Fig. 1. Illustration of major extreme points, (left) minimum and (right) maximum

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 major extreme points.

Given a time series T, starting at the beginning of the time series, all major minima and maxima of the time series are computed by using the algorithm given by Pratt and Fink [14]. The algorithm takes linear time and constant memory.

PLA Segmentation using the Sliding Window Algorithm. The approximation of a time series T, of length m, with K straight lines that tightly fit the original data points is the essence of Piecewise Linear Approximation (PLA). The algorithms which input a time series and returns a piecewise linear representation are called *PLA segmentation algorithms*. The PLA segmentation problem can be formulated as follows. Given a time series, scan and divide the entire time series into a number of segments such that the

maximum error for any segment does not exceed some user-specified threshold, *max_error*.

According to Keogh et al. [5], most time series PLA segmentation algorithms can be classified into three categories: Sliding Window, Top-Down and Bottom-Up. In this work, we select to use Sliding Window algorithm for PLA segmentation due to its procedural simplicity and its online nature.

3 Improving the HOT SAX Algorithm

In this work, we attempt to improve the HOT SAX algorithm through the following three techniques:

- Estimating the suitable size of PAA frame based on PLA segmentation.
- Estimating the suitable value for discord length n and SAX word length w based on identifying major extreme points.
- Applying a new way of shifting the sliding window.

So, our improved HOT SAX is a two-pass approach. In the first pass, we estimate the size of PAA frame, the discord length n and the word length w. Then, we apply HOT SAX with a new way of shifting sliding window and subsequence distance computation.

3.1 An Efficient Way to Estimate the Size of PAA Frame

The use of PAA representation in HOT SAX is mainly for dimensionality reduction. But in [6], Keogh et al. have not mentioned how much dimensionality reduction we should do for a given time series in order to guarantee the accuracy of discord discovery. It is obvious that too much dimensionality reduction may cause the loss of information in the original time series and this can be harmful to the accuracy of discord detection. Therefore, the size of PAA frame, which indicates how much dimensionality reduction we want, should be considered as an important parameter in HOT SAX and we should have a principled method to estimate it.

In this work, we propose a technique for estimating the size of PAA frame which is based on PLA segmentation. We select PLA segmentation for estimating the size of PAA frame due to the following reason. Because of the linear representation bias, PLA segmentation algorithms are much more effective at producing fine grain partitioning, rather than a smaller set of segments that represent characteristic patterns in a time series.

Among three algorithms for PLA segmentation: Top-Down, Bottom-Up and Sliding Window, Sliding Window algorithm is selected. After applying Sliding Window algorithm to divide the time series into linear segments, we can set the PAA frame for dimensionality reduction equal to the average length of all PLA segments obtained.

Notice that the complexity of the Sliding Window algorithm for PLA segmentation is just linear [5] and the estimation of the parameter *max_error* in the Sliding Window algorithm is intuitive. If the time series is very complex, we would favor a fine grain partitioning, which corresponds to a small PAA frame, then a small *max_error* is a good

choice. Otherwise, relatively smooth and slowly changing datasets favor a larger value of *max_error*.

3.2 An Efficient Way to Estimate Discord Length and SAX Word Length

Identifying major extreme points [14] is a good method for segmentation that can extract characteristic patterns in a time series. Applying this method, we assume that any candidate patterns for discord discovery must start from a major extreme point and ends at the next major extreme point. With this assumption, we can estimate the discord length for a given time series by the following scheme. After identifying all the major extreme points in the time series, we extract subsequence segments one by one from each pair of adjacent major extreme points in the time series. We keep track of the lengths of all extracted subsequence segments. Next, we set the discord length equal to the average length of all the extracted subsequence segments.

For the algorithm that identifies all the major extreme points in the time series, we have to determine the compression rate R. According to [14], Pratt and Fink only suggested that R should be greater than 1. In this work, to find the right value for R, we have to try some values for R in the range from 1 to 10 and identify the value which seems to bring out an appropriate segmentation for a particular time series. Thank to the ease of visualization of segmented time series, we can estimate the right value for R with not much effort.

Furthermore, from the discord length n and the size of the PAA frame, we can easily determine the SAX word length w for HOT SAX algorithm by the following formula.

$$w = n/(\text{size of PAA frame})$$

3.3 A New Way to Shift the Sliding Window

The original HOT SAX creates a SAX representation of the entire time series by sliding a window of length n across the time series T one data point at a time. Hence, the original HOT SAX is computational expensive. A question can be raised "Is it necessary to shift the sliding window only one data point at a time?". To our knowledge, several previous works did not apply the same way of shifting sliding window as in HOT SAX. Chuah and Fu (2006) [2] proposed an adaptive window-based discord discovery (AWDD) approach for anomaly detection in ECG time series. AWDD shifts the sliding window across the ECG time series from a peak to its adjacent peak at a time. Tanaka et al. (2005) [16] proposed EMD, a time series motif discovery algorithm which also applies PAA and SAX transformation. EMD shifts the sliding window, called the *analysis window*, across the time series one PAA frame at a time. Following the spirit from EMD algorithm, in this work, we select to shift the sliding window one PAA frame at a time in order to speed-up the discord discovery process.

3.4 Other Issue: How to Compute Subsequence Distance in Improved HOT SAX

In HOT SAX, the conversion of subsequences to SAX words in this algorithm implies the use of the MINDIST distance between two SAX words, which is given in [11]. Besides this way to compute the distance between two SAX words, we can refer back to the two corresponding subsequences of these two SAX words in the original time series and compute the Euclidean distance between them. The latter way of computing subsequence distance can bring out a better accuracy of discord discovery than the former one.

4 Experimental Evaluation

We implemented all three algorithms, Brute Force, HOT SAX and Improved HOT SAX. The experiments aim to compare Improved HOT SAX with original HOT SAX in terms of time efficiency and discord detection accuracy. The Bruce Fore is used as the baseline algorithm.

Our experiments were conducted over the datasets from the UCR Time Series Data Mining archive for discord discovery [7]. There are 11 datasets used in these experiments. The datasets are from different areas (finance, medicine, manufacturing, science). After applying PLA segmentation to estimate the size of PAA frame and identifying major extreme points to estimate the discord length n for each dataset, we obtained the two important parameters for each dataset as shown in Table 1. We set the threshold *max_error* in the range from 0.3 to 0.000002.

Table 1. Discord length and size of PAA frame for each dataset

Dataset	Length	*max_error*	*R*	Discord length (n)	Size of PAA frame
218c3EEG	8500	0.00007	1.499998	62	2
stock_20_0	5000	0.006	1.500001	138	3
memory	6000	0.000003	1.500000	357	3
ECG	6000	0.0025	1.499990	183	3
Power_demand_italy	6000	0.015	1.499997	267	3
ERP	5545	0.08	1.500014	164	2
chromosome	6000	0.0009	1.499996	99	3
eeg	6000	0.3	1.499996	63	3
koski_ecg	10000	0.00001	1.499999	633	3
power	5000	0.0015	1.499994	234	2
stock	6000	0.00002	1.499992	1410	3

4.1 Accuracy

Table 2 shows the experimental results about the discords found on each of 11 time series datasets by the three algorithms: Brute Force, HOT SAX and Improved HOT SAX. Over 11 datasets, we found out that the discord detected by Improved HOT SAX or HOT SAX is exactly the same as the discord detected by Brute Force. However, for each dataset there is some difference between the start location of the discord found by the HOT SAX or Improved HOT SAX to that of the same discord found by Brute Force.

From Table 2, we found out that the location difference between Brute Force and Improved HOT SAX is always much smaller than the location difference between Brute Force and HOT SAX. This phenomenon indicates that our Improved HOT SAX can detect time series discords better than the original HOT SAX.

Table 2. Locations of discords detected by Brute-Force, HOT SAX and Improved HOT SAX

Dataset	*n*	Position of discord			Location differences	
		BFDD	HOT SAX	Improved HOTSAX	HOT SAX vs. BFDD	Improved HOT SAX vs. BFDD
218c3EEG	62	6052	6145	6057	93	5
stock_20_0	138	97	4862	97	4765	0
memory	357	5217	3012	5197	2205	20
ECG	183	4015	4083	4015	68	0
Power_demand_italy	267	5323	2504	5323	2819	0
ERP_data	164	2551	2671	2551	120	0
chromosome	99	428	1973	430	1545	2
Eeg	63	3082	3019	3079	63	3
koski_ecg	633	8583	8393	8584	190	1
power	234	4615	41	4615	4574	0
Stock	1410	1	4544	1	4543	0

4.2 Efficiency

Following the tradition established in [1, 6], the efficiency of the three algorithms is measured by the number of calls to the distance function. Table 3 shows the numbers of distance function calls by Bruce Force, HOT SAX and Improved HOT SAX.

Table 3. Numbers of distance function calls by Bruce Force, HOT SAX and Improved HOT SAX

Dataset	The number of distance function calls		
	BFDD	HOTSAX	Improved HOTSAX
218c3EEG	70182506	6504476	481691
stock_20_0	22330350	2135119	383117
memory	27957656	3025101	328580
ECG	31758860	1785801	278003
Power_demand_italy	29893556	9165021	846848
ERP	27232742	1857263	511564
chromosome	33680612	205302	28455
Eeg	34521500	2689901	138630
koski_ecg	76308960	13903056	1219958
power	20552622	2506976	864376
Stock	10121942	2373381	203773

Experimental results in Table 3 show that HOT SAX brings out about 1 order of magnitude of a speedup to Brute Force while Improved HOT SAX has about 2 orders of magnitude of a speedup to Brute Force. Additionally, for sanity check, we measured the execution times of the three algorithms over 11 datasets. The experimental results in Table 4 reveal that in average, HOT SAX can work faster than Brute-Force about 4 times and Improved HOT SAX runs faster than HOT SAX about 2.5 times.

Table 4. Execution times (in seconds) of Brute-Force, HOT SAX and Improved HOT SAX

Dataset	Runtime (sec)		
	BFDD	HOTSAX	Improved HOTSAX
218c3EEG	66.023	22.153	7.266
stock_20_0	45.785	11.115	5.923
memory	143.013	16.611	7.131
ECG	84.032	15.566	6.394
Power_demand_italy	115.062	29.671	8.145
ERP	67.370	24.336	10.80
chromosome	50.372	15.011	7.119
eeg	34.104	11.263	3.473
koski_ecg	678.439	104.546	22.212
power	70.261	17.256	10.214
stock	203.061	41.343	12.42

We also conducted an experiment to compare the performance of HOT SAX in two ways of computing subsequence distances: MINDIST on SAX words and Euclidean Distance between two subsequences in original time series. The results of this experiment shows that the location difference between the discord detected by Brute Force and the discord detected by HOT SAX using MINDIST is always greater than the location difference between the discord detected by Brute Force and the discord detected by HOT SAX using Euclidean distance. These results indicate that HOT SAX with Euclidean distance on subsequences performs better than HOT SAX with MINDIST on SAX words.

5 Conclusions

In this paper, we proposed some new techniques to improve HOT SAX algorithm in time series discord discovery. This work has two major contributions. Firstly, we mitigate the difficulty of setting two important parameters, discord length n and word length w of HOT SAX by replacing them with the two other easy parameters, *max_error* and R in the two time series segmentation methods: major extreme points and PLA segmentation. Secondly, we speed-up HOT SAX by shifting the sliding window with one PAA frame at a time rather than one data point at a time.

In the future, we plan to devise some other method for outer and inner heuristics in HOT SAX for more effectiveness and efficiency.

References

1. Bu, Y., Leung, T.W., Fu, A., Keogh, E., Pei, J., Meshkin, S.: WAT: Finding top-K discords in time series database. In: Proceedings of the 2007 SIAM International Conference on Data Mining (SDM 2007), Minneapolis, MN, USA, 26–28 April 2007
2. Dani, M.C., Jollois, F.X., Nadif, M., Freixo, C.: Adaptive threshold for anomaly detection using time series segmentation. In: Arik, S., Huang, T., Lai, W.K., Liu, Q. (eds.) ICONIP 2015, Part III. LNCS, vol. 9491, pp. 82–89. Springer, Switzerland (2015)
3. Fu, T.C., Chung, F.L., Ng, C.M.: Financial time series segmentation based on specialized binary tree representation. In: Proceedings of 2006 International Conference on Data Mining, pp. 3–9 (2006)
4. Gruber, C., Coduro, M., Sick, B.: Signature verification with dynamic RBF network and time series motifs. In: Proceedings of 10th International Workshop on Frontiers in Hand Writing Recognition (2006)
5. Keogh, E., Selina, C., David, H., Michel, P.: An online algorithm for segmenting time series. In: Proceedings of the IEEE International Conference on Data Mining, pp. 289–296 (2001)
6. Keogh, E., Lin, J., Fu, A.: HOT SAX: efficiently finding the most unusual time series subsequence. In: Proceedings of 5th ICDM, Houston, Texas, pp. 226–233 (2005)
7. Keogh, E.: www.cs.ucr.edu/~eamonn/discords/. (Accessed on 24 Jan 2015)
8. Kha, N.H., Anh, D.T.: From cluster-based outlier detection to time series discord discovery. In: Li, X.L., Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D. (eds.) PAKDD 2015. LNCS (LNAI), vol. 9441, pp. 16–28. Springer, Switzerland (2015)
9. Leng, M., Chen, X., Li, L.: Variable length methods for detecting anomaly patterns in time series. In: International Symposium on. Computational Intelligence and Design (ISCID 2008), vol. 2 (2008)
10. Li, G., Braysy, O., Jiang, L., Wu, Z., Wang, Y.: Finding time series discord based on bit representation clustering. Knowl.-Based Syst. **52**, 243–254 (2013)
11. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: Symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, San Diego, CA, 13 June 2003
12. Lovric, M., Milanovic, M., Stamenkovic, M.: Algorithmic methods for segmentation of time series: an overview. JCEBI **1**(1), 31–53 (2014)
13. Oliveira, A.L.I., Neto, F.B.L., Meira, S.R.L.: A method based on RBF-DAA neural network for improving Novelty detection in time series. In: Proceedings of 17th International FLAIRS Conference. AAAI Press, Miami Beach (2004)
14. Pratt, K.B., Fink, E.: Search for patterns in compressed time series. Int. J. Image Graph. **2**(1), 89–106 (2002)
15. Salvador, S., Chan, P.: Learning states and rules for time series anomaly detection. Appl. Intell. **23**(3), 241–255 (2005)
16. Tanaka, Y., Iwamoto, K., Uehara, K.: Discovery of time series motif from multi-dimensional data based on MDL principle. Mach. Learn. **58**, 269–300 (2005)