Feature-based Online Segmentation Algorithm for Streaming Time Series (Short Paper)

Peng Zhan[1], Yupeng Hu[1(✉)], Wei Luo[1], Yang Xu[2], Qi Zhang[1], and Xueqing Li[1(✉)]

[1] School of Software, Shandong University, Jinan, Shandong, China
{huyupeng,xqli}@sdu.edu.cn
[2] School of Computer Science and Technology, Shandong University, Qingdao, Shandong, China

Abstract. Over the last decade, huge number of time series stream data are continuously being produced in diverse fields, including finance, signal processing, industry, astronomy and so on. Since time series data has high-dimensional, real-valued, continuous and other related properties, it is of great importance to do dimensionality reduction as a preliminary step. In this paper, we propose a novel online segmentation algorithm based on the importance of TPs to represent the time series into some continuous subsequences and maintain the corresponding local temporal features of the raw time series data. To demonstrate the advantage of our proposed algorithm, we provide extensive experimental results on different kinds of time series datasets for validating our algorithm and comparing it with other baseline methods of online segmentation.

Keywords: Data mining · Streaming time series · Online segmentation · Algorithm

1 Introduction

Nowadays, a great number of intelligent devices in extensive fields are continuously producing streaming time series. Due to the high-dimensional, large amount, continuous and other related properties, it is unrealistic to do further data mining research on the raw streaming time series directly. Accordingly, the online segmentation for streaming time series should be done to reduce both the space and the computational cost in the first place. Online segmentation approach for streaming time series, which is aimed for not only providing the segmentation continuously, but also ensure the corresponding results retain the main temporal features of the raw data.

– The similarity-based matching and pattern recognition of time series data first need to discover several "primitive shapes" [1] or "frequent patterns" [2] subsequences, which can be used for reducing the complexity of the following similarity measurement steps.

H. Gao et al. (Eds.): CollaborateCom 2018, LNICST 268, pp. 477–487, 2019.
https://doi.org/10.1007/978-3-030-12981-1_33

- In time series outlier detection tasks, the segmentation approach can improve the efficiency of eliminating those normal time sequences.
- The typical prototypes [3] should be created for the predefined classes when doing time series classification tasks, which could be generated by the segmentation approach as a preprocessing step.
- In time series clustering tasks, meaningful temporal feature time sequences instead of raw data points, produced by segmentation process, would improve the convergence ability [4] of clustering algorithms, such as K-means.

In this paper, we aim to produce approximate representation which contains the main temporal features and process streaming time series efficiently. The Corresponding contributions could be summarized as follows.

1. We propose an online segmentation algorithm for streaming time series, called feature-based online segmentation algorithm (FOS), which subdivides the time series by a set of TPs, which do reflect the corresponding local temporal features.
2. We evaluate the different importance of TPs by standing on a more holistic view and selecting the most important TP to preform backward and forward segmentation to maintain a high similarity between the series of segments and the raw time series.
3. Comprehensive experiments have been conducted on both open source and different types of the UCR time series datasets in comparison with other baseline methods to demonstrate the advantages of our proposed algorithm.

2 Related Work

Scholars have already proposed a large number of time series segmentation methods, including Discrete Fourier Transform (DFT) [5,6], Discrete Wavelet Transform (DWT) [7], Singular Value Decomposition (SVD) [8], Piecewise Aggregate Approximation (PAA) [9], Piecewise Linear Representation (PLR) [10] and Symbolic Aggregate approXimation (SAX) [11].

Piecewise Linear Representation (PLR) refers to the approximation of a time series T, of length n, with K connected straight lines [12], which is in line with human visual experience, and lower dimension and faster calculation speed of PLR makes the storage, transmission of the data more efficient [1,3,13]. Therefore, PLR is more suitable for segmenting and approximatively representing streaming time series. The segmentation based on PLR can be described as follow:

For a given time series $T = (a_1, a_2, \ldots, a_i, \ldots, a_j, \ldots, a_n)$ of length n, where n can be a constant value or continues to grow without restriction. The T will be divided into sequences $S = (S_1, S_2, \ldots, S_k)$ while $(1 \leq k \leq n-1)$ and be represented by a series of connected straight lines. Overall, PLR methods can be categorized into two classes: Offline PLR and Online PLR, which is an important distinction because many data mining applications in real world are inherently dynamic. In general, the offline PLR methods segment the whole time series

data sequence, and the online PLR methods segment data sequences based on the data seen so far.

2.1 Offline PLR

Offline PLR algorithms segment the whole time series data sequence, that is to say, the whole datasets should be collected before PLR conducting. Although these Offline PLR methods have different implementation details, they can be grouped into one of the following two categories.

- Top-Down algorithm based on PLR (PLR-TD) [14]: The main idea of PLR-TD is considering every possible partitioning of the data sequence and segmenting at the best location according to some user-specified threshold. Both subsegments are then verified to see whether their representation error is below the threshold, if not, PLR-TD recursively splits the subsegments until the stopping criterion is met.
- Bottom-Up algorithm based on PLR (PLR-BU) [15]: PLR-BU is the natural complement to PLR-TD. PLR-BU begins with $n-1$ segments, and then it begins to iteratively merge the lowest cost two adjacent segments until a stopping criteria is met.

2.2 Online PLR

Onine PLR algorithms mainly concentrate on PLR for streaming time series, which segment data sequences based on the data seen so far. The upcoming data sequence will be acquired and piecewisely approximated at the same time rather than gathering the whole data sequence at the very beginning.

One classic Online PLR algorithm is Slide Windows algorithm based on PLR (PLR-SW). It works by initializing the first data point of time series as the initial segmentation point (i.e., the left endpoint) of a segment and trying to find the right endpoint of the segment by put one more data point into the segment in each step [16]. Still and all, the main problem of Online PLR methods is lacking the global view of its offline counterparts, so that the fitting error of segmentation is usually less than satisfactory. To solve this problem, Keogh et al. [12] introduces an approach in which they hold the online nature of Sliding Windows (SW) and retain the superiority of Bottom-Up (BU) called SWAB (**S**liding **W**indow **a**nd **B**ottom-Up) to improve the fitting precision of SW method. In order to improve computing efficiency, Liu et al. [17] proposes a new concept of segmentation criterion called feasible space to reach the farthest segmenting point of each segment, and then introduces two Online PLR methods, which are the Feasible Space Window (FSW) method and the Stepwise Feasible Space Window (SFSW) method. These two methods greatly improve the computing efficiency of the segmentation, but the fitting error is larger than SWAB in most cases due to the lack of overall understanding on the temporal characteristics.

According to the comparative analysis of the existing classic Online PLR algorithms, Online PLR methods are able to do continuous segmentation, but

the major drawback of them is the fitting precision of representation can not be guaranteed, compared to the Offline PLR methods.

3 Preliminary

3.1 The Definition of Turning Points

In order to divide the streaming time series T into continuous segments more reasonably, it is preferable to find those segmenting data points of special significance, in other words, the approximate straight lines by connecting those segmenting data points would reserve the variation trend features of raw time series. In this paper, those data points denote the variation trends, called the Turning Points (TPs) will be defined in Definition 1.

Definition 1 (TP). *For a streaming time series TS with n data points, which is growing continuously, could be expressed as $T = \{a_1, a_2, \ldots, a_{i-1}, a_i, a_{i+1} \ldots, a_n\}$, where $1 \leq i \leq n$, element $a_i = (t_i, v_i)$ indicates the recorded value v_i arrives at the distinct timestamp t_i. Considering the time-order in T is obvious ($i < i+1$), which could be simplified as $T = \{v_1, v_2, \ldots, v_{i-1}, v_i, v_{i+1} \ldots, v_n\}$. If v_i satisfies one of the following two inequations, v_i can be defined as a TP.*

$$v_{i-1} < v_i > v_{i+1} \text{ or } v_{i-1} < v_i = v_{i+1} \text{ or } v_{i-1} = v_i < v_{i+1} \tag{1}$$

$$v_{i-1} > v_i < v_{i+1} \text{ or } v_{i-1} > v_i = v_{i+1} \text{ or } v_{i-1} = v_i > v_{i+1} \tag{2}$$

According to the above definition, all the TPs in T could be found completely. In order to make Definition 1 more intuitively, time series T of GunPoint dataset [18] has been taken for example and shown in Fig. 1. In this Figure, it is not difficult to find that all the local extreme points, such as TP_2, TP_3, and the inflection point (TP_5) have already been selected as TPs in T. However, each TP in T does has different degree of importance in T, in other words, each TP has a disparate contribution to preserve the local temporal features and maintain the global temporal trend of T. Therefore, it is necessary to sort all the TPs based on their own TP importance (TPI) from high to low, preparing for the subsequent online segmentation. Without loss of generality, according to the previous research work [19], the traditional vertical distance (VD) and the mean value of the current T could be utilized to sort the TPI of TPs in T orderly. The definition of TPI is described as follow.

Definition 2 (TPI). *For a streaming time series TS, the mean value of T, named MT, could be calculated by the following equation. The vertical distance (VD) between TP_i in T and MT could be defined as the importance of TP_i, denoted as TPI_i. When TP_i has the maximum VD(MVD), TP_i is the most important TP in T.*

$$MT = \frac{\sum_{i=1}^{N} v_i}{N} \tag{3}$$

$$TPI_i = \sqrt{(v_i - MT)^2} \tag{4}$$

In Fig. 1, all the TPIs of TPs in T have been obtained by calculating VD in Eq. 4. TP_5 with the maximum VD (1.8) could be identified as the most important TP in T.

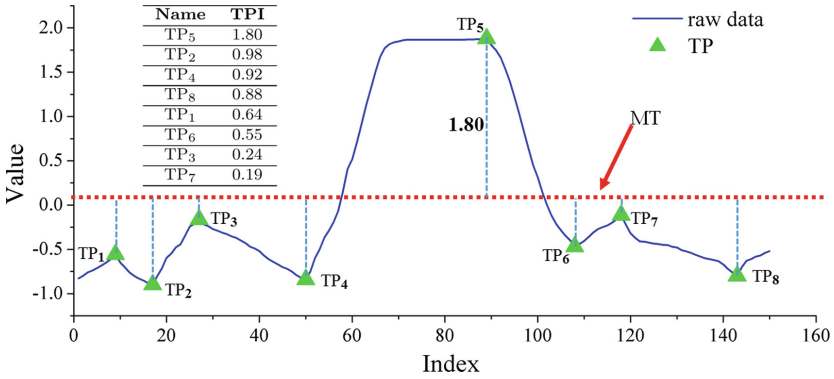


Fig. 1. Intuitive example of TPs definition and importance

3.2 The Slope Calculation and Segmentation Criteria

To ensure the fitting precision of FOS, a segmentation criterion need to be introduced for a potential segment, which we call the Maximum Error for Single Point (ME_SP). ME_SP is used to evaluate the fitting error of a single data point in a potential segment. In order to improve the computing efficiency, the FSW method subsitute the slope calculation (SC) for the calculation of the maximum vertical distance (MVD) [17]. Similarly, in our algorithm, we adopt slope calculation to speed up segmentation, and combine with finding TPs. Table 1 illustrates some definitions of the slope calculation, as follow.

In particular, for a time series T, whose ME_SP could be specified by users and be denoted as σ, where $\sigma > 0$.

Table 1. The Definitions of slope calculation

Name	Description
$line(a_i, a_j)$	The straight line connected by point a_i and a_j
$sline(a_i, a_j)$	The slope of the straight line connected by point a_i and a_j
$slow(a_i, a_j)$	The slope of the straight line connected by point a_i and $a_i - \sigma$
$sup(a_i, a_j)$	The slope of the straight line connected by point a_i and $a_i + \sigma$

If $sline(a_i, a_k)$ satisfies the Inequation 5, the MVD between a_j and $line(a_i, a_k)$ will not exceed the σ(ME_SP). According to the segmentation criteria, every time a new data point arrives, we simply compare the slope of the new

line $sline()$ with $slow()$ and $sup()$ of the current segment and update them to find $maxslow_{(i:j)}$ (the maximum value of $slow()$) and $minsup_{(i:j)}$ (the minimum value of $sup()$). The corresponding equations are listed in Eqs. 6 and 7.

$$slow(a_i, a_j) \leq sline(a_i, a_k) \leq sup(a_i, a_j) \tag{5}$$

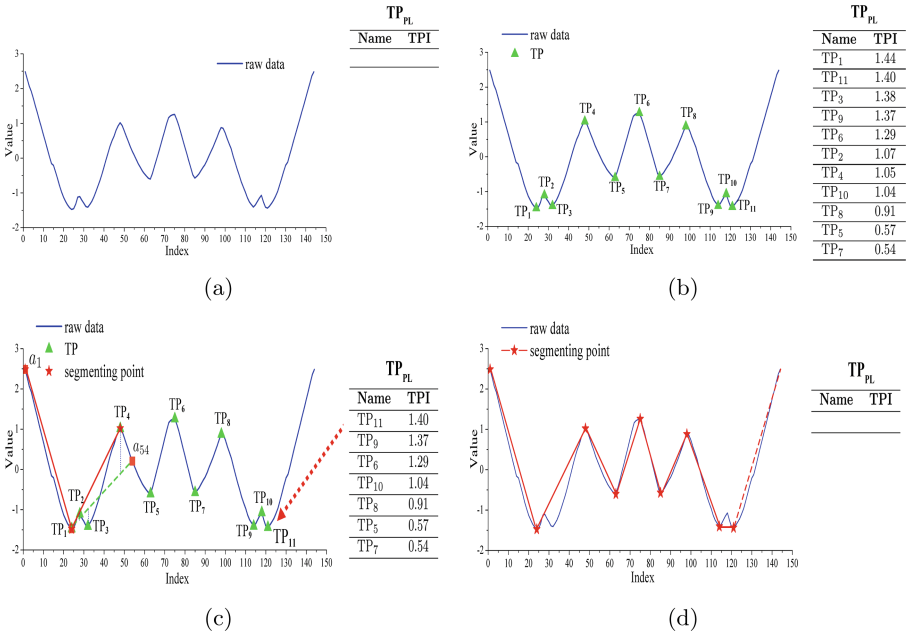
$$maxslow_{(i:j)} = max_{i<t<j} slow(a_i, a_t) \tag{6}$$

$$minsup_{(i:j)} = min_{i<t<j} sup(a_i, a_t) \tag{7}$$

The current segmentation will not end until such condition:$maxslow_{(i:j)} > minsup_{(i:j)}$ is satisfied, and then to repeat the above operation from the current segmenting point until the entire streaming time series data has been processed.

4 Algorithm

According to the above definitions, FOS can be subdivided into two major steps, as follow.



TP_{PL}	
Name	TPI

TP_{PL}	
Name	TPI
TP_1	1.44
TP_{11}	1.40
TP_3	1.38
TP_9	1.37
TP_6	1.29
TP_2	1.07
TP_4	1.05
TP_{10}	1.04
TP_8	0.91
TP_5	0.57
TP_7	0.54

TP_{PL}	
Name	TPI
TP_{11}	1.40
TP_9	1.37
TP_6	1.29
TP_{10}	1.04
TP_8	0.91
TP_5	0.57
TP_7	0.54

TP_{PL}	
Name	TPI

Fig. 2. The major steps of FOS

4.1 TP Selection and Evaluation

In the beginning, a buffer named buf should be created for storing the upcoming steaming time series T, and the setting of buf is same as SWAB [12]. Along with the data of T is constantly flowing into buf, TP in T could be identified according to the definition of TP in Sect. 3. After the buf is full, all the TPs in buf would be sorted based on their own degree of importance from high to low. In order to illustrate the above process more clearly, the TPs selection and evaluation on time series in Plane dataset [18] has been taken for example and shown in Fig. 2. Figure 2(a) and (b) demonstrate this process in detail. The green triangle points denote all the TPs in current buf and all the TPs have also been sorted in accordance with their own TPI from high to low and stored into the TP priority list, named TP_{PL}, in Fig. 2(b).

4.2 Adaptive Stepwise Segmentation by SC and TPS

After the above process, the most important TP (TP_1) in TP_{PL} has been identified in Fig. 2(b). Different from traditional FSW, SFSW and SWAB, which treat each data point in T equally and subdivide T in buf from the starting point, FOS performs SC-based forward and backward segmentation operation from TP_1. Due to TP_1 is selected from buf by standing on the "global" view, the corresponding forward and backward segmentation operation will become more meaningful. As shown in Fig. 2(c), the SC-based forward and backward segmentation operation would start at TP_1, the backward segmentation ends at point a_1 and the forward segmentation ends at point a_x. To preserve the basic trend of time series more rationally, the forward segmentation between TP_1 and point a_x, as shown in green dotted line in Fig. 2(c), would be refined by measuring the VD of TP_2, TP_3 and TP_4 mentioned in Sect. 3. After the corresponding calculation, TP_4 would be selected as the final segmenting point in the current forward segmentation, shown in red dotted line in Fig. 2(c), TP_1, TP_2, TP_3 and $TP4$ would be removed from TP_{PL}. Subsequently, the current most important TP (TP_{11}) in TP_{PL}, should be selected and the corresponding forward and backward segmentation would be performed at TP_{11}. So circulates, until TP_{PL} is empty. The final segmentation in current buf is shown in Fig. 2(d). After the final segmentation in current buf has been finished completely, except the rightmost subsegment, as shown in red dotted line in Fig. 2(d), all the subsegments would be removed from buf and the new follow-up data of T would flow into buf for the next segmentation until the entire T has been processed completely.

According to the final segmentation result in current buf, it is obviously that all the subsegments are formed by TPs, which could not only preserve the basic trend of time series more rationally, but also maintain high degree of similarity between the processed segments and the raw data sequence. The corresponding comparison and explanation would be given in detail in Sect. 5.

5 Experiment and Analysis

5.1 Dataset and Evaluation Metrics

In order to perform the experiment, we select some kinds of typical time series datasets of different fields, including finance, signal processing, industry provided by UCR Time Series Archive [18], and we also choose some representative industrial streaming time series including the monitoring data of Dong Fang Hong satellite (DFHS) from January 2015 to June 2015, which is the Chinese satellite dataset provided by China Academy of Space Technology, the Hang Seng Index (HSI) from 4th January 2016 to 30th December 2016 from the Yahoo Financial web site, and the monitoring data of Jinan municipal steam heating system(JMSHS) from December 2014 to March 2017.

Since our algorithm choose those important TPs as potential segmenting points, which can reflect the variation trend of the data sequence, the similarity between approximation representation and the raw data is relatively high. That is to say, each subsegment could not only reflect the basic trend of time series intuitively, but also minimize the holistic representation error as much as possible. Therefore, we use the representation error (RE) based on ME_SP to evaluate the performance of segmentation for streaming time series.

5.2 Comparison with Baseline Methods on Representation Error

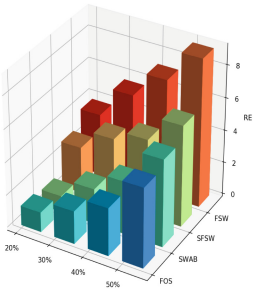
In order to compare the corresponding segmentation performance more objectively, we compare FOS with three baseline methods, which are SWAB, FSW and SFSW, using the identical ME_SP. To make the experimental results more credible, we define two conditions in advance before the experiments.

For one thing, we adopt the Maximum Error Percentage for Single Point (MEPP) proposed by Liu et al. [17], which can reduce the sensitivities of different time series datasets to ME_SP by choosing appropriate ME_SP values for each dataset. For another, the results of SWAB with MEPP are considered as the benchmark (i.e., each result of SWAB is set as 1), and then we normalize the results of other methods according to the benchmark method. Table 2 illustrates the normalized representation error (NRE) results of four Online PLR methods on the above listed datasets.

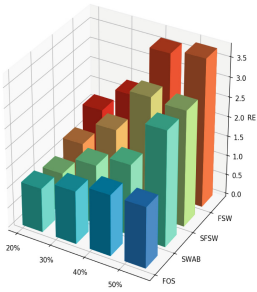
In Table 2, the NRE results of SWAB on the above datasets are set as the benchmark (1), whose original NRE results are also listed in this Table. By comparing the specific NRE between FOS and FSW, SFSW on all the above datasets, it is not difficult to find that the NRE results of FOS are much smaller than that of FSW and SFSW by its global feature-based segmentation. Through the comparison between FOS and SWAB on the same datasets, we could also find that all the NRE results of FOS are smaller that of SWAB on all the datasets except DFHS. Through the corresponding analysis on DFHS, we find that the NRE of FOS is affected by the distribution of TPs in some cases. For instance, due to the concentrated distribution of TPs in DFHS dataset, the relatively more important TPs would be removed from the TP_{PL} in the previous forward

Table 2. NRE results of four methods

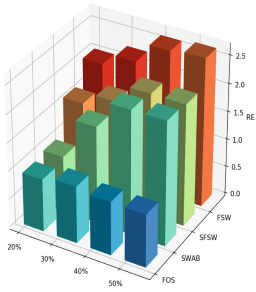
Dataset	SWAB	FSW	SFSW	FOS
Plane	**1** (28.83)	2.63	1.89	0.94
Mallat	**1** (298.72)	1.56	1.29	0.89
Strawberry	**1** (124.09)	0.85	1.05	0.39
OSULeaf	**1** (73.67)	2.99	2.04	0.97
Car	**1** (113.71)	2.67	1.58	0.57
JMSHS	**1** (786.32)	2.43	2.39	0.89
DFHS	**1** (8751.86)	2.21	2.17	1.11
HSI	**1** (96632.69)	1.64	1.21	0.59
Average	**1**	2.12	1.70	0.79



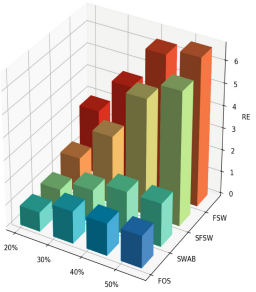
(a) NRE results on Plane



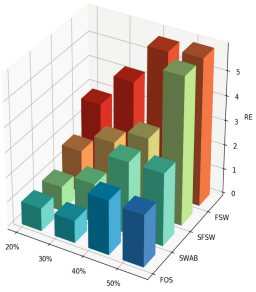
(b) NRE results on Mallat



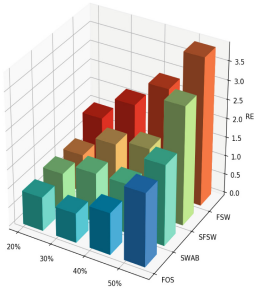
(c) NRE results on Strawberry



(d) NRE results on OSULeaf



(e) NRE results on Car



(f) NRE results on HSI

Fig. 3. The comparison of NRE on different datasets

and backward segmentation, and the subsequent operations could only perform the corresponding segmentation based on TPs with a relatively less importance, which would lead to the NRE result of FOS is bigger than SWAB. Finally, according to the average NRE results of the above four methods on all the

datasets, it is obvious that FOS can provide more accurate segmentation than other three methods in general.

In order to further analyze the segmentation performance of FOS on different datasets, extensive experiments would be conducted on the above 6 datasets by varying the MEPP from 20% to 50%. Moreover, In order to distinguish the differences between the four methods more clearly, the NRE results on these 6 datasets based on MEPP (20%) by SWAB are set as the benchmark (1) and the corresponding normalized NRE results of the four methods on 6 datasets have been shown in Fig. 3. In this figure, it is not difficult to find that NRE results on all the datasets rise gradually along with MEPP increases. Moreover, due to the lack of a more comprehensive view of segmentation, the NRE results of FSW and SFSW are much larger than SWAB and FOS in general. Although both SWAB and FOS perform online segmentation from a global perspective, SWAB treats all data points equally, while FOS selects the current most important TP to preform backward and forward segmentation. Therefore, the NRE result of FOS is much smaller than that of other three methods, in other words, a relative high similarity between the approximate representation and the raw data could be maintained by FOS.

6 Conclusion

In this paper, we propose a novel feature-based online segmentation algorithm (FOS) which reserves the main characteristic of time series and performs well on diverse streaming time series datasets. The extensive experimental results demonstrate that FOS can guarantee more accurate approximate representation of streaming time series. In future, we plan to use FOS for time series retrieval and anomaly detection.

References

1. Chiu, B., Keogh, E., Lonardi, S.: Probabilistic discovery of time series motifs. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 493–498. ACM (2003)
2. Lonardi, J., Patel, P.: Finding motifs in time series. In: Proceedings of the 2nd Workshop on Temporal Data Mining, pp. 53–68 (2002)
3. Bagnall, A., Lines, J., Hills, J., Bostrom, A.: Time-series classification with COTE: the collective of transformation-based ensembles. IEEE Trans. Knowl. Data Eng. **27**(9), 2522–2535 (2015)
4. Fayyad, U.M., Reina, C., Bradley, P.S.: Initialization of iterative refinement clustering algorithms. In: KDD, pp. 194–198 (1998)
5. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57301-1_5
6. Rafiei, D., Mendelzon, A.: Efficient retrieval of similar time sequences using DFT. arXiv preprint cs/9809033 (1998)

7. Chan, F.K.-P., Fu, A.W.-C., Yu, C.: Haar wavelets for efficient similarity search of time-series: with and without time warping. IEEE Trans. Knowl. Data Eng. **15**(3), 686–705 (2003)
8. Ravi Kanth, K.V., Agrawal, D., Singh, A.: Dimensionality reduction for similarity searching in dynamic databases. In: ACM SIGMOD Record, vol. 27, pp. 166–176. ACM (1998)
9. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowl. Inf. Syst. **3**(3), 263–286 (2001)
10. Keogh, E., Chu, S., Hart, D., Pazzani, M.: Segmenting time series: a survey and novel approach. In: Data Mining in Time Series Databases, pp. 1–21. World Scientific (2004)
11. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A 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, pp. 2–11. ACM (2003)
12. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: Proceedings IEEE International Conference on Data Mining, ICDM 2001, pp. 289–296. IEEE (2001)
13. Hu, Y., Ji, C., Jing, M., Li, X.: A K-motifs discovery approach for large time-series data analysis. In: Li, F., Shim, K., Zheng, K., Liu, G. (eds.) APWeb 2016. LNCS, vol. 9932, pp. 492–496. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45817-5_53
14. Keogh, E.J., Pazzani, M.J.: An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In: KDD, vol. 98, pp. 239–243 (1998)
15. Park, S., Lee, D., Chu, W.W.: Fast retrieval of similar subsequences in long sequence databases. In: Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange (KDEX 1999), pp. 60–67. IEEE (1999)
16. Qu, Y., Wang, C., Wang, X.S.: Supporting fast search in time series for movement patterns in multiple scales. In: Proceedings of the Seventh International Conference on Information and Knowledge Management, pp. 251–258. ACM (1998)
17. Liu, X., Lin, Z., Wang, H.: Novel online methods for time series segmentation. IEEE Trans. Knowl. Data Eng. **20**(12), 1616–1626 (2008)
18. Chen, Y., et al.: The UCR time series classification archive, July 2015. www.cs.ucr.edu/~eamonn/time_series_data/
19. Si, Y.W., Yin, J.: OBST-based segmentation approach to financial time series. Eng. Appl. Artif. Intell. **26**(10), 2581–2596 (2013)