Dynamic Adaptive Piecewise Linear Representation Approach Based on Streaming Time Series

Wei Luo¹, Peng Zhan¹, Qi Zhang¹, Cun Ji², Jiecai Zheng³,a,* and Xueqing Li¹,b,*
¹School of Software, Shandong University, Jinan, China
²School of Information Science & Engineering, Shandong Normal University, Jinan, China
³School of Sport Communication & Information Technology, Shandong Sport University, Jinan, China
aEmail: jczheng@sdu.edu.cn
bEmail: xqli@sdu.edu.cn
*Corresponding Author

Abstract. With the burgeoning of IoT (Internet of Everything), massive numbers of IoT devices in extensive fields are continuously producing huge number of streaming time series. The high dimensionality and dynamic uncertainty of this kind of data lead to the main challenge on traditional time series data mining research. Accordingly, time series representation methods have been regarded as a necessary pre-processing tool to provide data support for the follow-up time series data mining research. In this paper, we propose a novel dynamic time series representation approach called dynamic adaptive piecewise linear representation (DAPLR) for streaming time series, which can automatically provide a series of piecewise linear representation results to meet the diverse needs of different users.

1. Introduction

Along with the proliferation of IoT, the deployment of extensive IoT instruments (sensors etc.) in various industry fields to produce huge number of streaming time series (STS). More importantly, STS could be regarded as an ordered collection of elements, in which new elements are continuously generated potentially forever [1]. Therefore, it is limited capability to reveal the regularity and uncertainty of STS by traditional similarity measured methods, which used to process on static time series datasets.

Accordingly, dimensionality reduction and features representation become crucial to efficient data mining and knowledge discovery, since it reduce both capacity and computational complexity of storing and transmitting time series data[2]. There are several highly cited representation methods [3], including Discrete Fourier Transform (DFT)[4], Discrete Wavelet Transform (DWT)[5], Singular Value Decomposition (SVD)[6], Piecewise Aggregate Approximation (PAA)[7], Piecewise Linear Representation (PLR)[8] and Symbolic Aggregate aproXimation(SAX)[9].

In these traditional time series representation methods, piecewise linear representation (PLR) is more widely used for its simplicity, high efficiency and more suitable for human visual intuition [10].

Although traditional PLR methods could complete corresponding representation for streaming time series, the results by these methods have been fixed on the user-specified constraints, in other words, the representation results could not be dynamically adjusted according to the changing needs of users.

Considering this situation, in this paper, we propose a novel dynamic time series representation approach called dynamic adaptive piecewise linear representation (DAPLR) method for streaming
time series, which can provide a series of time series representation results to meet the diverse needs of different users. The subsequent experiments have been conducted to demonstrate that DAPLR can provide the corresponding results to meet the diverse needs of different users without unnecessary repeated calculation.

2. Methodology

In this section we will describe our proposed dynamic adaptive piecewise linear representation (DAPLR) method based on important data points (IDPs) in detail. First of all, different representation criteria and the corresponding definitions on representation for streaming time series would be given, and then the overview of DAPLR would be introduced, finally the specific process for time series based on different methods and the relevant index construction are described respectively.

2.1. Definitions of Time Series and Representation Criteria

**Definition 1 (TS):** For a time series with \( n \) points could be expressed as \( TS = \{p_1, p_2, \ldots, p_1, p_{i_1}, \ldots, p_n\} \), where \( 1 \leq i \leq n \), element \( p_i = (v_i, t_i) \) indicates the recorded value \( v_i \) arrives at the distinct timestamp \( t_i \), and \( n \) denotes a positive integer, which could continue to grow without limited. Without loss of generality, supposing a sliding window (SW) with length \( m \) should be adopted for receiving the subsequences of \( TS \), expressed as \( S_j = \{S_1, S_2, \ldots, S_j, \ldots, S_m\} \), where \( j \) denotes the \( j \)th subsequence generated in SW. \( S_j \) could be further expressed as \( S_j = \{p_{j(i-m+1)}, \ldots, p_j, \ldots, p_{j(m)}\} \).

When we plan to segment time series sequences into some continuous piecewise linear representation, it is advisable for us to segment the sequences according to their temporal features. The essence of the fitting precision is to make sure the fitting lines match their original temporal features as closely as possible, in other words, the distance between the raw data sequence and the fitting line is expected to be as small as possible.

According to the above analysis, vertical distance is used to represent fitting error in this paper. The cumulative fitting error (vertical distance) between the raw sequence data in one segment and its fitting line is much larger than in other segments, the weight of segment (WoS) can be supposed to be much higher than in other segments, in other words, WoS is closely related to the cumulative fitting error, the larger cumulative fitting error is, the higher WoS will be.

According to Definition 1, a series of piecewise representation results could be generated by PLR based on IDPs and the corresponding representation criteria. Therefore the definition of IDP and the representation criteria are given as follows.

The important data points (IDPs) can be considered as the core reference condition of dynamic adaptive representation. We can also use IDPs to divide an original high weight segment into two new low weight segments. Therefore, the distance between the raw data sequence and the fitting line would be closer than before. The definition of IDPs can be described as follows.

**Definition 2 (IDP):** For a subsequence \( S_j = (a_1, \ldots, a_i, \ldots, a_m) \) in the current SW, the first data point \( a_1 \) and the last data point \( a_m \) should be defined as IDPs. Moreover, if \( (a_1, \ldots, a_i) \) in \( S_j \) is formed a segment named \( A \), whose weight is larger than other segments. The point \( a_x \) in segment \( A \), whose single point fitting error, named as \( \text{distsum} \), is larger than other points in \( A \) would be selected as an IDP.

**Definition 3 (FER):** FER is aimed for dynamic adaptive representation, which is the ratio of the current entire fitting error to the maximum entire fitting error. The definition of FER can be expressed as equation (1).

\[
\text{FER} = \frac{f_{e_{\text{cur}}}}{f_{e_{\text{max}}}} \quad (1)
\]

In equation (1), \( f_{e_{\text{cur}}} \) is the current entire fitting error and \( f_{e_{\text{max}}} \) is the maximum entire fitting error which can also be described as the initial fitting error shown in figure 1(a).
In figure 1(a), the first and the last data point have been selected as IDPs for linear fitting and $\text{dist}_{\text{sum}}$ in the initial segmentation indicated by the green line is much larger than any other situations. The largest $\text{dist}_{\text{sum}}$ can be set as the benchmark (set as 1), so we can normalize the fitting error in other situations with the benchmark. With the orderly selection of other IDPs, the $\text{dist}_{\text{sum}}$ (fitting error) would gradually decrease. If all the data points have been selected as the IDPs for segmentation, $\text{dist}_{\text{sum}}$ would become zero.

The flexible implement of adaptive representation will be facilitated by FER, which can meet any requirements based on the corresponding fitting error rate.

**Definition 4 (DCR):** DCR is similar to FER, which is the ratio of the current number of IDPs to all selected data points in dataset. DCR can be expressed as equation (2).

$$DCR = \frac{\text{num}_{\text{cur}}}{\text{num}_{\text{max}}}$$

(2)

In equation (2), $\text{num}_{\text{cur}}$ is the current number of IDPs and $\text{num}_{\text{max}}$ is the number of all selected data points in time series. We can take figure 1(d) as an example, $\text{num}_{\text{cur}}$ is 5 and $\text{num}_{\text{max}}$ can be supposed as 20, so DCR is 0.25 ($\frac{5}{20}$).

### 2.2. Dynamic Adaptive Representation Algorithm

Figure 1 shows the main steps in the piecewise linear representation based on IDPs, which can be classified into the following two categories:

1. The piecewise linear representation with IDPs by comparing WoS based on $\text{dist}_{\text{sum}}$. In this situation, the cumulative fitting error of the whole segment exceeds the pre-specified error threshold, the single point with the maximum fitting error should be chosen as the new segmentation point to split the initial segment into two new segments, as shown in figure 1 (a) and (b).

2. The piecewise linear representation with IDPs by comparing WoS based on $n\times\text{dist}_{\text{max}}$. In this situation, the cumulative fitting error of the entire segment does not exceed the pre-specified error threshold, but $n\times\text{dist}_{\text{max}}$ in this segment exceeds the error threshold. In order to avoid the fitting error of single point is too large, the data point whose error is $\text{dist}_{\text{max}}$ should be chosen as the new segmentation point to split the initial segment into two new segments, as shown in figure 1 (c) and (d).

In our algorithm, all IDPs should be defined as TreeNode before inserted into the dynamic adaptive index structure. The object TreeNode is expressed as $\text{TreeNode} = \{\text{index}; \text{value}; \text{rank}; \text{error}_L; \text{error}_R; \text{child}_L; \text{child}_R\}$, and the attributes of TreeNode are described in table 1.
Table 1. Attributes of TreeNode

| Attributes | Description                                      |
|------------|--------------------------------------------------|
| index      | The original position of the point.              |
| value      | The value of the point.                          |
| rank       | The order in which the point is selected.        |
| error_L    | The fitting error of the left segment when the point has been selected. |
| error_R    | The fitting error of the right segment when the point has been selected. |
| child_L    | The left child of this TreeNode.                 |
| child_R    | The right child of this TreeNode.                |

Objects of Tree: All TreeNode are planned to put into the index tree, and the object Tree is defined as tree = {root}. The root is an object of TreeNode.

Objects of Segment: Segment would record the WoS and the index of the IDP. Therefore, Segment is expressed as Segment = \{p_b, p_e, p_max, w\}. The attributes of Segment are described in table 2. The index tree construction algorithm is shown in algorithm 1.

Table 2. Attributes of Segment

| Attributes | Description                                      |
|------------|--------------------------------------------------|
| p_b        | The begin point of the segment.                  |
| p_e        | The end point of the segment.                    |
| p_max      | The point which has maximum distance to fitting line of the segment |
| S_weight   | The weight of the segment.                       |

In Line 1 of algorithm 1, we initialize a list list1 to record all segments that are generated in the calculation process. From line 2 to line 5, we put the first and end points of time series in the index tree. MAX is the maximum value of type double. Lines 10 to 26 repeat a cycle of operations:

1) Obtain the Segment S1 from list using the function getSegment().
2) Divide S1 into two segments and determine whether the newly generated segments should be put into list according to the actual situation.
3) Insert the point S1.p_max into the tree.
4) Remove the segment that has been divided and sorted in list.
Algorithm 1 Tree Construction Algorithm

**Input:** time series data: \( x = (x_1, x_2, \ldots, x_n) \)

**Output:** the binary tree: \( \text{tree} \)

1: List \( \text{list} = \text{new ArrayList}(); \)
2: TreeNode \( \text{root} = \{1, x_1, 1, \text{MAX}, \text{MAX}, \text{null}, \text{null}\}; \)
3: Tree \( \text{tree} = \text{new Tree}(\text{root}); \)
4: Segment \( \text{segment} = \text{getSegment}(1, n); \)
5: TreeNode \( \text{treeNode} = \{n, x_n, 2, \text{segment.w}, \text{MAX}, \text{null}, \text{null}\}; \)
6: if \( \text{segment.w} > 0 \) then
7: \( \text{list.add(segment)}; \)
8: end if
9: \( \text{rank} = 3; \)
10: while \( \text{null} \neq \text{list} \&\& \text{list.size()} > 0 \) do
11: \( \text{Segment segL = list.get(0)}; \)
12: \( \text{Segment segR = getSegment(seg1.pb, seg1.pmax)}; \)
13: if \( \text{segL.w} > 0 \) then
14: \( \text{list.add(segL)}; \)
15: end if
16: \( \text{Segment segR = getSegment(seg1.pmax, seg1.pe)}; \)
17: if \( \text{segR.w} > 0 \) then
18: \( \text{list.add(segR)}; \)
19: end if
20: TreeNode \( \text{treeNode1} = \{pmax, x_{pmax}, \text{rank}, \text{segL.w}, \text{segR.w}, \text{null}, \text{null}\}; \)
21: \( \text{tree.insert(treeNode1)}; \)
22: \( \text{list.remove(0)}; \)
23: \( \text{list.sortByWeight()}; \)
24: \( \text{rank}++; \)
25: end while
26: \( \text{return tree} \)

We repeat the above four steps until there is no segment in \( \text{list1} \). Finally, in line 27, the algorithm returns the index tree of the time series data set. The information for each node is given in Table 1.

To the best of our knowledge, in a large number of time series data mining works, the demands of the fitting error on the same time series are often very different, even in the same research, the demands in different stages could vary widely. Besides, the requirements of the number of points for representing the same time series are also very different in some kinds of data regression analysis and data visualization works. DAPLR is different from other traditional PLR methods which can only represent a time series with an unchangeable fitting error and a fixed number of segmentation points. In other words, DAPLR can provide a more flexible representation based on different FER and DCR, which can be used by different time series data analysis and mining tasks (similarity search, pattern recognition, visualization, classification, clustering, etc.) as the pre-processed step. The dynamic adaptive representation strategy of DAPLR can be divided into the two main categories as follows.

1) **dynamic adaptive representation based on DCR**

Some users specify a certain number of segments to represent the time series, in other words, DCR is also specified at the same time. The algorithms that return a specific number of IDPs are shown as algorithm 2 and algorithm 3. Algorithm 2 is the entry function for the dynamic adaptive representation based on DCR. In this algorithm, the list for storing IDPs and the root of the index tree should be set at the beginning and then algorithm 3 would be invoked to find IDPs. Algorithm 3 is a recursive procedure, which is similar to the in-order traversal of index tree. The entire traversal search process starting at the root node will continue until the rank attribute of the selected tree node is greater than \( DCR \times N \) (\( N \) is the number of data point of the original time series).
Algorithm 2 selectIDPsByNum

**Input:** the number of IDPs to return: \( k \), the binary tree: \( \text{tree} \)

**Output:** the IDPs list \( IDPs \)

1: List \( IDPs = \text{new ArrayList()} \);
2: TreeNode \( \text{treenode} = \text{tree.root} \);
3: \( \text{selectIDPsByNum}(k, \text{treenode}, \text{IDPs}) \)
4: return \( IDPs \)

Algorithm 3 selectIDPsByNum

**Input:** the number of IDPs to return: \( k \), tree node: \( \text{treeNode} \), the list to record data: \( IDPs \)

1: if \( \text{null} == \text{treeNode} || \text{treeNode.rank} > k \) then
2: return
3: end if
4: \( \text{selectIDPsByNum}(k, \text{treenode.childL}, \text{IDPs}) \)
5: \( \text{IDP}.\text{add}({\text{treenode.index, treenode.value}}) \)
6: \( \text{selectIDPsByNum}(k, \text{treenode.childR}, \text{IDPs}) \)

2) dynamic adaptive representation based on FER

Some users expect the fitting precision for all the segments can be limited in an acceptable range, in other words, FER is also specified simultaneously. The algorithm 4 and algorithm 5 return appropriate piecewise linear representations, which are similar with the algorithm 2 and algorithm 3. The only difference is that entire traversal search process will continue until the fitting error ratio for all segments is smaller than FER.

Algorithm 4 selectIDPsByWeight

**Input:** the threshold: \( \varepsilon \), the binary tree: \( \text{tree} \)

**Output:** the IDPs list \( IDPs \)

1: List \( IDPs = \text{new ArrayList()} \);
2: TreeNode \( \text{treenode} = \text{tree.root} \);
3: \( \text{selectIDPsByWeight}(\varepsilon, \text{treenode}, \text{IDPs}) \)
4: return \( IDPs \)

Algorithm 5 selectIDPsByWeight

**Input:** the threshold: \( \varepsilon \), tree node: \( \text{treeNode} \), the list to record data: \( IDPs \)

1: if \( \text{null} == \text{treeNode} \) then
2: return
3: end if
4: if \( \text{treeNode.errorL} > \varepsilon \) then
5: \( \text{selectIDPsByWeight}(\varepsilon, \text{treenode.childL}, \text{IDPs}) \)
6: end if
7: \( \text{IDP}.\text{add}({\text{treenode.index, treenode.value}}) \)
8: if \( \text{treeNode.errorR} > \varepsilon \) then
9: \( \text{selectIDPsByWeight}(\varepsilon, \text{treenode.childR}, \text{IDPs}) \)
10: end if

3. Applications

Through the above analysis, DAPLR can provide a more flexible representation to meet different needs of users. To make our point of view more clearly, we can take the dynamic adaptive representation for part of Longda as an example. DAPLR can represent the same time series sequence with different fitting error based on FER, shown in figure 2. Analogously, DAPLR can also represent the same time series sequence with different number of IDPs based on DCR to satisfy the different needs of visualization. As shown in figure 3, users can flexibly select proper DCR to represent the same time series in different form according to different display devices.
4. Conclusion
In this paper, we propose a novel dynamic representation method DAPLR. DAPLR performs well on producing adaptive piecewise linear representation results for different users without unnecessary repeated calculation. In future, we plan to use DAPLR as a useful tool for the follow-up time series data mining research.

5. References
[1] Y. Hu, P. Ren, W. Luo, P. Zhan, and X. Li, “Multi-resolution representation with recurrent neural networks application for streaming time series in IoT,” Computer Networks, 2019.
[2] Y. Hu, C. Ji, M. Jing, and X. Li, “A k-motifs discovery approach for large time-series data analysis,” in International Conference on Asia-Pacific Web Conference.
[3] Y. Hu, P. Guan, P. Zhan, Y. Ding, and X. Li, “A novel segmentation and representation approach for streaming time series,” IEEE Access, 2018.
[4] R. Agrawal, C. Faloutsos, and A. Swami, “Efficient similarity search in sequence databases,” in International Conference on Foundations of Data Organization and Algorithms. Springer, 1993, pp. 69–84.
[5] F.-P. Chan, A.-C. Fu, and C. Yu, “Haar wavelets for efficient similarity search of time-series: with and without time warping,” IEEE Transactions on knowledge and data engineering, vol. 15, no. 3, pp. 686–705, 2003.
[6] K. Ravi Kanth, D. Agrawal, and A. Singh, “Dimensionality reduction for similarity searching in dynamic databases,” in ACM SIGMOD Record, vol. 27, no. 2. ACM, 1998, pp. 166–176.

[7] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, “Dimensionality reduction for fast similarity search in large time series databases,” Knowledge and information Systems, vol. 3, no. 3, pp. 263–286, 2001.

[8] E. Keogh, S. Chu, D. Hart, and M. Pazzani, “Segmenting time series: A survey and novel approach,” in Data mining in time series databases. World Scientific, 2004, pp. 1–21.

[9] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, “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. ACM, 2003, pp. 2–11.

[10] Y. Hu, C. Ji, M. Jing, Y. Ding, S. Kuai, and X. Li, “A continuous segmentation algorithm for streaming time series,” in International Conference on Collaborative Computing: Networking, Applications and Worksharing. Springer, 2016, pp. 140–151.