A shape-based adaptive segmentation of time-series using particle swarm optimization

Article in Information Systems - July 2017
DOI: 10.1016/j.is.2017.03.004

3 authors, including:

Hossein Kamalzadeh
Southern Methodist University
1 PUBLICATION 1 CITATION

Abbas Ahmadi
Amirkabir University of Technology
51 PUBLICATIONS 209 CITATIONS

Some of the authors of this publication are also working on these related projects:

- Proposing Intelligent Cooperative System Based on Classifier Ensemble for Lung Cancer Diagnosis View project
- Multiple cooperative swarms clusering View project
A shape-based adaptive segmentation of time-series using particle swarm optimization

Hossein Kamalzadeh, Abbas Ahmadi *, Saed Mansour
Department of Industrial Engineering & Management Systems, Amirkabir University of Technology, Tehran, Iran

**Abstract**

The increasing size of large databases has motivated many researchers to develop methods to reduce the dimensionality of data so that their further analysis can be easier and faster. There are many techniques for time-series’ dimensionality reduction; however, majority of them need an input by the user such as the number of segments. In this paper, the segmentation problem is analyzed from the optimization point of view. A new approach for time-series’ segmentation based on Particle Swarm Optimization (PSO) is proposed which is highly adaptive to time-series’ shape and shape-based characteristics. The proposed approach, called Adaptive Particle Swarm Optimization Segmentation (APSOS), is tested on various datasets to demonstrate its effectiveness and efficiency. Experiments are conducted to show that APSOS is independent of user input parameters and the results indicate that the proposed approach outperforms common methods used for the time-series segmentation.

© 2017 Elsevier Ltd. All rights reserved.

1. Introduction

Nowadays, by the use of advanced technologies for data stor- age, large amounts of data from various application domains such as finance and stock market, medicine and health care, bioinform-atics and biology [1], telecommunication and telemetry, sensor networking, motion tracking, speech recognition [2], genetics, mul-timedias, meteorology [1,3], astronomy [4–6], robotics, industry [7], business and social sciences [8], are being constantly and continu-ously saved into databases for further analysis. These analyses may include common methods in data mining domain such as cluster-ing, classification, mining of association rules and pattern recogni-tion [9,10]. Usually this data are stored in the form of time-series, which is a sequence of real numeric values recorded or sampled at continuous time steps with either same or different sample rates [2]. These datasets are usually massive with sometimes more than a half terabyte of data and are updated with gigabytes of data ev-eryday [4,9,11]. As it seems, for two major reasons, mining of this data is difficult; first, this data is changing over time thus clas-sic data mining methods which are useful for static data may not work well enough with the streaming data [12]. Second reason is related to the high dimensionality of data, one of the most im-

* Corresponding author at: Room 624, Department of Industrial Engineering & Management Systems, Amirkabir University of Technology, 424 Hafez Ave., Tehran, Iran.

E-mail addresses: farzadkamalzadeh@aut.ac.ir (H. Kamalzadeh), abbas.ahmadi@aut.ac.ir (A. Ahmadi), s.mansour@aut.ac.ir (S. Mansour).

http://dx.doi.org/10.1016/j.ins.2017.03.004 0305-4479/C

2017 Elsevier Ltd. All rights reserved.
optimization perspective for the segmentation problem and explains the problem in the de- sired form. In Section 4, a new technique for dimensionality reduc- tion based on PSO and optimization perspective is proposed and later in Section 5 the proposed method is compared to some of the existing methods in the area. Finally, Section 6 concludes the paper with an overview of the proposed method's superiorities to other techniques.

2. Related works

Dimensionality reduction techniques for time-series are divided into various categories based on the type of algorithm used and how they try to reduce the dimension of a given time-series. In [15] these techniques are divided into 2 major categories called data adaptive and non-data adaptive. The data adaptive methods are those which segment a time-series using varied length seg- ments; but in non-data adaptive, each time-series is divided into a number of segments with the same length. This taxonomy of rep- resentation methods is completed in [1] by putting the represen- tation methods into four categories: data adaptive, non-data adapt- ive, model-based and data dictated. In model-based approaches the time-series is represented by a stochastic model and in data dictated methods contrary to the other three categories, the con- pression ratio is figured automatically from the time-series. In this paper, the focus is on data adaptive and non-data adaptive adap- tive methods, since the proposed method works by the raw time-series in the form of numeric values not a model.

Among the data adaptive methods, the most efficient ones are Piecewise Linear Approximation (PLA) [16], Adaptive Piecewise Constant Approximation (APCA) [9], Derivative Segment Approxi- mation (DSA) [2], Symbolic Aggregate Approximation (SAX) [3] and Singular Value Decomposition (SVD) [17], and among the non-data adaptive ones, Piecewise Aggregate Approximation (PAA) [4], Dis- crete Wavelet Transform (DWT) [18] and Discrete Fourier Trans- form (DFT) [19] are the most efficient. These techniques can be divided into two groups based on the approaches they take to- ward the dimensionality reduction goal [2]. Some of these meth- ods reduce the dimension of a time-series through a continu- ous function which is usually linear. This group includes PLA, APCA, DSA, PAA and SAX. The other group, considers a low- order continuous function for each time series and then selects ad- equate coefficients to best approximate the time-series. This group includes SVD, DFT and DWT. It is notable to say that the first group is widely known as the segmentation methods and the goal in segment- ing a time series is to reduce the associated dimensionality by omitting as many points as possible while preserving the underlying characteristics such as trends, local extremums, turning points or inflection points. In other words, these methods are shape-based dimensionality reduction. They try to save the shape of a time-series while sacrificing its points.

Many factors are important in time-series segmentation as it is a method which causes loss of data. First, the more it is inde- pendent of the user input parameters the better and easier it is to use. Most of the methods require the number of segments in advance and estimating the number of segments for each time- series in a large database is a really time consuming task. Second, while the segmentation causes a loss of data it should save the shape and relevant features of the time-series. Finally the trans- formed time-series should be able to be represented again in the time domain and also can be handled by some powerful similar- ity measures such as dynamic time warping [20]. It must be noted that the benefit of using piecewise discontinuous approximations is that they can be combined with existing similarity measures. Therefore, the computational cost of similarity search decreases with a great amount [2].

PLA [16] selects a number of points from the original time- series. By matching these points a set of linear segments are gen- erated instead of the original time-series. Thus, the outcome is the set of the endpoints building linear segments. Although PLA is a fast and accurate method, it needs an initial solution: the number of segments and the points to be preserved. For the initial solution, integral square error (ISE) approximation is used in which the sum of squares of the pointwise errors is taken as the error norm. Al- though, this method provides an initial solution, it still needs an input parameter which is the error threshold. This is the main dis- advantage of the PLA technique. In this paper the ISE approxima- tion is also used to provide the initial solution for not only the PLA but also for those methods which need the number of segments as an input.

PAA [4] and APCA [9] are like each other in the approach they take toward reducing the dimensionality of a given time-series. They both transform a time-series into a number of segments whose values are the mean value of the points between the two endpoints of the segment. The outcome in these methods is a set of pairs of the beginning points and the mean values. The differ- ence is that in PAA the segments have the same length but in APCA the segments can be of different lengths, which in result makes the APCA more accurate than PAA. Moreover, these methods need the number of segments as an input parameter.

DSA [2] has a different way of reducing the dimension. It first estimates the first derivative of the points in a given time-series based on a model similar to the one proposed in [21] and then segments the time-series by aggregating subsequent data points whose derivatives are very close. Therefore, the outcome will be a new sequence whose elements are pairs of time steps and an- gles. The advantage of DSA is that it is independent of any user input parameter but it usually yields to low compression ratios.

In this paper, four of the techniques in the literature including PLA, PAA, APCA and DSA are selected to be compared with the pro- posed algorithm. The main reason for choosing these techniques is that these methods take a discontinuous approach toward dimen- sionality reduction. In other words, they are segmentation methods and their outcomes have almost the same form. The second reason is that according to the literature these methods have accurate results for the goal of time-series segmentation [1,15].

Since many symbols and notations are used in this paper some of which are specific to this paper, for the sake of simplicity, im- portant ones are explained in Table 1.

3. Optimization form of segmentation

The segmentation problem has been viewed from various per- spectives such as dynamic programming [22,23], balanced error so- lutions [24], variance minimization and discrete optimization [16]. Among the four perspectives, the discrete optimization one is more popular as it is easier to understand and formulate. Even in this perspective, various approaches can be considered, but the main goal usually is to minimize an objective function equal to the er-
Table 1
Symbols used and their definitions.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_S$</td>
<td>Time-series</td>
<td>A sequence of real numeric values recorded or sampled at continuous time steps</td>
</tr>
<tr>
<td>$STS$</td>
<td>Segmented Time-series</td>
<td>A time-series which is a shorter-length form of a raw time-series</td>
</tr>
<tr>
<td>$N$</td>
<td></td>
<td>Number of time steps in a time-series</td>
</tr>
<tr>
<td>$K$</td>
<td></td>
<td>Number of segments (or roughly the number of points)</td>
</tr>
<tr>
<td>$ε_k$</td>
<td>Error</td>
<td>Error of each point</td>
</tr>
<tr>
<td>$m_k$</td>
<td>Indices of the time-series’ points</td>
<td></td>
</tr>
<tr>
<td>$P$</td>
<td>Particle $p$</td>
<td>A point in a space representing a solution to the optimization problem (in PSO)</td>
</tr>
<tr>
<td>$P$</td>
<td></td>
<td>Number of particles (swarm’s population)</td>
</tr>
<tr>
<td>$X_p$</td>
<td>Position vector of particle $p$</td>
<td>A vector of $K$ elements which acts as a solution to the optimization problem</td>
</tr>
<tr>
<td>$V_p$</td>
<td>Velocity vector of particle $p$</td>
<td>A vector of the same size as position vector which is used to change the position vector elements</td>
</tr>
<tr>
<td>$X_{pb}$</td>
<td>Particle $p$’s best position</td>
<td>The best solution particle $p$ has ever had</td>
</tr>
<tr>
<td>$w$</td>
<td>Inertia weight</td>
<td>The weight of the particle’s last velocity vector in the new velocity vector</td>
</tr>
<tr>
<td>$c_1$</td>
<td>Cognitive coefficient</td>
<td>A coefficient for the particle’s best move in the new velocity vector</td>
</tr>
<tr>
<td>$c_2$</td>
<td>Social coefficient</td>
<td>A coefficient for the swarm’s best move in the new velocity vector</td>
</tr>
<tr>
<td>$r_1$, $r_2$</td>
<td>Random values</td>
<td>Random values between $0$ and $1$</td>
</tr>
<tr>
<td>$v_c$</td>
<td>Velocity constraint coefficient</td>
<td>A coefficient used to control the velocity upper bound value</td>
</tr>
<tr>
<td>$t_{er}$</td>
<td>Iteration number of algorithm</td>
<td>Iteration number of algorithm</td>
</tr>
<tr>
<td>$ε_1$, $ε_2$</td>
<td>Threshold values</td>
<td>Threshold values for the standard deviation of last solutions</td>
</tr>
<tr>
<td>$δ_1$, $δ_2$</td>
<td>Number of iterations</td>
<td>for which the standard deviation of the solutions is compared to a specific threshold</td>
</tr>
<tr>
<td>$k$</td>
<td>Initial value for the number of segments</td>
<td></td>
</tr>
</tbody>
</table>

![Fig. 1. Original time-series and the segments representing it.](image)

The time-series $T_S = \{x_1, x_2, \ldots, x_N\}$ is a time-series of length $N$ when the index $i$ is the time step of the value $x_i$. The goal of dimensionality reduction is to reduce the number of data-points or time steps of the given time-series. More specifically, the goal of segmentation here is to segment the time-series into $K$-$1$ linear segments which in fact needs $K \leq N$ points of the original time-series to be identified as the delimiters or endpoints of the segments. By considering the fact that there is no constrain on the length of the segments and they can be varied, the new time-series, hereafter called the Segmented Time-Series ($STS$), is a set of $K$ points from the original time-series, given by $STS = \{u_1, u_2, \ldots, u_{K}\}$, when $u_i \in T_S$. This is represented in Fig. 1 as an example.

In Fig. 1 a time-series of length 91 is represented with 6 segments, and in order to best represent the given time-series, 7 points are selected in a way that its characteristics and features such as local extremums and trends can be preserved. Although, a big amount of data has been removed, the new time-series still inherits the original one’s shape and characteristics. The time-series value for each time steps of the original time-series can be estimated roughly from the relevant line whose accuracy is defined by its two endpoints. To get the approximate value, each point of time-series should be mirrored vertically on the line segment and the value of that point on the line will be the approximate value. But as $K \leq N$, and now only the linear segments are approximating the whole original time-series, as it is obvious from Fig. 1, there is a difference between the original time-series value for each time steps and its approximation. This difference is error and can be formulated. Consider $T_S = \{i, x_i\}$ when $i = 1, 2, \ldots, N$ and the approximation of $T_S$ from linear segments is $F = \{i, f_i\}$ when $i = 1, 2, \ldots, N$ and
\[ f_i = x^m = x^{-(i-n)} x_n, \quad (1) \quad mn \]

where \( n \leq i \leq m \), and \( x_n \) and \( x_m \) are respectively the left and right endpoints of the linear segment on which \((i, x_i)\) is vertically mirrored. Hence the error norm for each \((i, x_i)\) point can be calculated by

\[ e_i = \frac{1}{2} [x_i - f_i]^2, \quad (2) \]

where \( i = 1, 2, \ldots, N \) and \( e_i \) is the error norm for the point \((i, x_i)\).

If Eq. (2) is expanded on the \( i \), the total error will be

\[ E = \sum_{i=1}^{N} (x_i - f_i)^2. \quad (3) \]

If each \( f_i \) is replaced with its own equation as in Eq. (1), and \( m \), \( n \), and the indices of \( x_n \) and \( x_m \) update since they are segment-specific, total error function can be

\[ \sum_{i=m_2}^{m} \left( x_i - \frac{m_2 - m_1}{m_2 - m_1} (i - m_1) + x_{m_1} \right)^2, \quad \sum_{i=m_2}^{m} \]

\[ \sum_{i=m_2}^{m} \left( x_i - \frac{m_3 - m_2}{m_3 - m_2} (i - m_2) + x_{m_2} \right) + \sum_{i=m_2}^{m} \sum_{k=1}^{m_{k+1}} \left( \frac{x_i - m_{k+1}}{m_k - m_{k+1}} x_{m_k} - x \right) \quad \text{if } k \leq m, \quad (4) m_k \]

where \( k = 1, 2, \ldots, K, m_1 \in \{1, 2, \ldots, N\}, m_1 < m_{k+1}, m_1 = 1 \) and \( m_K = N \).

Eq. (4) can be represented in a simpler form as given below:

\[ E = \sum_{k=1}^{K} \left( x_{m_k+1} - \frac{m_k}{m_1} (i - m_1) + x_{m_1} \right)^2, \quad \sum_{k=1}^{K} \]

\[ \sum_{k=1}^{K} \left( x_{m_k+1} - \frac{m_k}{m_1} (i - m_1) + x_{m_1} \right) + \sum_{k=1}^{K} \sum_{m_k}^{m_{k+1}} \left( \frac{x_i - m_{k+1}}{m_k - m_{k+1}} x_{m_k} - x \right) \quad \text{if } k \leq m, \quad (5) \]

Based on the Eq. (5), the segmentation problem can be easily formulated as an integer programming problem (optimization problem) whose objective function is:

\[ \min_{m_k, x_{m_k}} \sum_{i=m_k}^{m_{k+1}} \left( x_i - \frac{m_{k+1} - m_k}{m} (i - m_1) + x_{m_1} \right)^2. \quad (6) \]

which is subject to \( k = 1, 2, \ldots, K, m_k \in \{1, 2, \ldots, N\}, m_k < m_{k+1}, m_1 = 1 \) and \( m_K = N \). As the variables of this problem are \( m_k \), the solution will be indices (time steps) of the points for the original time-series (TS) which have to be selected to reconstruct the TS or finally defined as STS. By finding these indices, the STS will be

\[ \{ u_1, u_2, \ldots, u_k, \ldots, u_k \}, \quad \text{when } u_k = TS(m_k) = x_{m_k}. \]

There are many algorithms that can solve an optimization problem including classics, heuristics and meta-heuristics, but in this paper PSO algorithm [25] is selected as it is a fast and accurate one specially for this case when the dimensionality of the data is very high [26].

The simple PSO generates a limited number of particles whose position vectors \( X_p = \{ m_1, m_2, \ldots, m_K \} = \{ m_k \} \) are each a solution to the optimization problem. Important thing about the position vector is that its size is fixed and known in advance. Each particle has also a velocity vector which helps it move toward a better solution. At every iteration, each particle’s position is given to the objective function and the returned value is compared to the last best position of the particle and the best one is saved as the per-sonal best position. Best personal- best of all the particles are compared and the best one is selected as the global best position. Particles’ velocities and positions are updated respectively by the following equations:

\[ V_{p_{iter}} = w V_{p_{iter}}^{\text{old}} + \text{c} F_1 (X_p - X_{G_{iter}}^{\text{old}}) \]

\[ \quad + \text{c} F_2 \left( X_G^{\text{best}_{iter-1}} - X_{p_{iter}}^{\text{old}} \right), \quad (7) \]

and

\[ X_{p_{iter}} = X_{p_{iter}}^{\text{new}} + V_{p_{iter}}, \quad (8) \]

where \( X_p^{\text{new}} \) is particle \( p \)’s next position, \( V_{p_{iter}}^{\text{new}} \) is particle \( p \)’s updated velocity, \( X_p^{\text{best}_{iter}} \) is particle \( p \)'s best position ever, \( X_G^{\text{best}_{iter}} \) is the global best position (swarm’s best position ever), \( r_1 \) and \( r_2 \) are uniformly ran-dom values between 0 and 1, \( w \) is the inertia weight and \( c_1 \) and \( c_2 \) are respectively the cognitive and social parameters. Through a number of iterations particles move toward the optimum solution which is the final global best position.

Using PSO for an integer programming problem needs some modifications on the algorithm [27].

First, PSO uses real values for particles’ positions and velocity vectors which in fact are useless when it is applied to an inter-ger optimization problem as the solution must be integer. To solve this problem a simple mapping should be applied on the parti-cles’ position vectors which maps each real number to an integer number in the desired period. In PSO, elements of position vector have a lower and upper bound which in simple form is respectively set to 0 and 1; i.e. the solution elements are all real values between 0 and 1. These numbers should be mapped to integer numbers between 1 and \( N \) (based on the constraint of Eq. (6)) which is

\[ m_1 \in \{1, 2, \ldots, N\} \]

\[ m_k = \{ N - 1 \} r_k + 1 \quad (9) \]

is used, where \( r_k \) is the \( k \)th element of the position vector which in fact is a real value between 0 and 1.

Second, since the position vectors update at each iteration, their values may go beyond the lower or upper bound and this causes a problem in mapping. Thus a constraint should be applied to the position vectors before mapping. Based on this constraint when an element of the position vector is larger than the upper bound, its value should be set equal to upper bound and when its value is smaller than the lower bound, it should be set equal to lower bound. By applying this constraint, position vectors are always in the desired interval.

Third, velocity vectors shouldn’t go beyond a certain limit as they will increase the position vectors’ values too much and these values will consequently go beyond the lower or upper bound; this way by applying the position constraint, position vectors’ values will always fall on the lower or upper bound. Hence, two other constraints are needed here to prevent this from happening. First, the velocity vector elements should have lower and upper bounds selected by

\[ V_{\text{Velocity Upper Bound}} = v_{c} \times \{ P \ osition \ Upper \ Bound \} \]
where $v_e$ is the velocity constraint coefficient. Second, when an element of velocity vector is larger than the upper bound, its value should be set equal to upper bound and when its value is smaller than the lower bound, it should be set equal to the lower bound. These constraints are called velocity constraints.

Finally, it is clear that if a position vector’s element goes beyond the lower or upper bound, and its respective velocity element stays in the same direction, by applying the position constraint, the position element will always stay on the bounds. To prevent this, for those position vector elements which are beyond the bounds, their respective velocity elements should be mirrored (multiplied by $-1$). This effect is called the velocity mirror effect.

By applying these modifications, PSO is now ready to solve the problem of segmentation which is now in the form of optimization.

4. Proposed algorithm: APSOS

In the last section, the segmentation problem was developed in the form of optimization and the goal was to minimize a total error by finding the indices of the given time-series points whose values can best approximate the time-series and save the shape of it. PSO was selected as the algorithm to solve the problem and essential modifications were applied to it. Running the algorithm for a time-series of length 4001, including 21 segments (estimated by user from the shape of the time-series) will give the result represented in Fig. 2.

As it is obvious from Fig. 2, most of the segments in early time steps are accurately adjusted but in late time steps, the segments are not sufficient to accurately represent the original time-series. The reason is that in simple PSO, the initial solution is uniformly distributed between 1 and 4001 but the local extremums in the original time-series are not uniformly distributed throughout the whole time-series, as in this example, the frequency is higher in the late time steps. Although in PSO particles move to find the best solution, their velocities become smaller and smaller through iterations as they try to search more nearby areas. This makes simple PSO incapable of solving the problem. Even after 1000 iterations, simple PSO cannot find the optimum solution for the example in Fig. 2.

Another drawback of using simple PSO for time-series segmentation is that in PSO the position vector size or the solution size (here $K$) is fixed and should be known and given to the algorithm in advance; i.e. the number of segments must be known as an input parameter. To solve these problems, the simple PSO algorithm is modified again and made adaptive to the time-series and the problem of segmentation. To adapt PSO to the segmentation problem, two major changes are applied to the algorithm, first on the objective function and second on the position vector size (solution size), and in result, the Adaptive Particle Swarm Optimization Segmentation (APSOS) algorithm is developed.

4.1. Objective function modification

The clear fact about PSO is that what controls the movement of the particles is the objective function, as it affects each particle’s cost, particles’ best costs and global best cost which all affect the velocities. Therefore, a proper objective function can easily enhance the final solution. The objective function in Eq. (6) can be represented in a simpler form as given below:

$$E = \sum_{k=1}^{K-1} e_k,$$

where $e_k$ is the total error for the $k$th segment. Doing so, the segments’ error for the example given in Fig. 2 is calculated and shown in Fig. 3. According to Fig. 3, segments 17, 19 and 21 produce a great amount of error, as the reason is clearly obvious from Fig. 2. But the rest of the segments have almost low amounts of error and they are almost around each other. It seems that those segments whose errors are more than the average error of all segments are not normal and the solution indicating this segmentation should be avoided. Thus the objective function must prevent other particles to approach this solution by increasing the error of the particle attaining this solution. Fortunately, PSO is a social cognitive algorithm; i.e., particles learn from each other and try to fol-

![Fig. 2. A time-series of length 4001 segmented into 21 segments by PSO.](image)
follow the solution whose segments’ errors are normal, around the average and small. This coefficient in APSOS will be called Segment Error Ratio (SER).

Another important fact is about the segments’ lengths. As a segment’s length increases, its error increases intrinsically. This fact can simply be understood if only one line is to approximate a whole time-series. Accordingly, particles should avoid such solutions in which segments’ lengths are too large. To do so another coefficient must be added to the objective function. The new objective function is

\[
E = \sum_{k=1}^{N} E_k(x) \times \mu_k \times \frac{L_{k+1}}{L_k}
\]

(13)

where like before \(m_{1,1}\) and \(m_{1,2}\) are respectively the right and left endpoint of the \(k\) th segment and \(N\) is the number of time steps (original time-series length). This fraction which is simply the ratio between the length of a segment and the length of the whole time-series will be called Segment Length Ratio (SLR) in APSOS.

\[
\text{Error}\text{ vs. Segments}
\]

**Fig. 3.** Segments’ errors for the time series of length 4091 with 21 segments.

### 4.2. Position vector modifications

These modifications are needed for two reasons. First, the SLR fraction encourages the particles to move toward shorter segments and therefore more number of points. But in fact the goal is to reduce the data size and this goal must be pursued as much as possible. Second, the algorithm must be free of input parameters such as number of segments or points. These two can both be reached by modifying the position vectors of particles. In simple PSO the position vector size is fixed and known in advance and doesn’t change through iterations, which makes this particular problem in need of the knowledge of the number of segments in advance. But in order to make the PSO in no need of this knowledge and adaptive to each time series and let the PSO estimate the number of segments by itself, position vector size should be changeable in APSOS. The mechanism works in this way that at the beginning, the algorithm starts with only three points (two segments) which are the first and last time steps of the original time-series and one point randomly selected between them; and through iterations the algorithm adds points to the specific segments to reduce their errors and relatively the total error. The mechanism should control when and where to add an extra point. One simple way is to add an extra point at each iteration but this way the algorithm and particles do not have enough time to search for the optimum solutions and the solution size changes and increases quickly. Thus every time the solution size increases by one point (one point is added) time in the form of iterations should be given to the algorithm to search for the optimum solution with the given number of points. In APSOS, if the standard deviation of the last step number of best global solutions is less than \(\varepsilon_1\), a new point is added. However, where to add this extra point is of great importance as this point should reduce the error as much as possible. The best choice is to add the extra point on the segment with the maximum error and somewhere randomly between its two endpoints so that the new point can break the segment line into two segments and reduce the total error. In Fig. 3, the next point should be added in between the two endpoints of segment 17. By adding an extra point, the size of the particles’ position vectors changes (increases) and therefore another element should also be added to the velocity vectors to comply with the position vector size.

The algorithm must also have stopping criteria. In APSOS, if the standard deviation of the last step number of best global solutions is less than \(\varepsilon_2\) and the number of points segmenting the time-series is at least equal to \(\kappa\), the algorithm will stop. These parameters are tuned in the next section of the paper.

Based on these modifications, the APSOS algorithm is represented in the pseudo code form in Fig. 4.

As mentioned before, \(E(.)\) is the objective function (error function) and parameters such as \(P, \varepsilon_1, \varepsilon_2, w, \varepsilon_3, \varepsilon_2, \delta_1, \delta_2, \kappa\) are APSOS specific parameters which are tuned in the next section of the paper.

For the parameter \(\kappa\), an algorithm based on Integral Square Error [16] is used which roughly estimates the number of segments, but this algorithm needs an input parameter as the threshold for the ISE error. In the next section this parameter is also tuned for every dataset and a universal value is provided.

An important fact about the outcome of APSOS is that time-series’ representation and shape do not change after being segment ment but the sample rate does. Since the segments have various lengths and delimiters can be selected anywhere from the given time-series, the segmented time-series is not evenly sampled. Although the outcome can be modified into an evenly sampled time-series by saving the lengths and the slopes, this doesn’t make any serious problems since this technique is mainly developed to provide shorter time-series for distance measures such as dynamic time warping which are capable of handling time-series with different sample rates.

### 5. Experimental results

In this section of the paper, the proposed algorithm is compared to four efficient methods called PLA, PAA, APCA and DSA on 9 datasets represented in Table 2 [28]. Each dataset consists of two sub-datasets separated as train and test. In this paper the test ones are used as they have more number of time-series. The reason for selecting these 9 datasets is that they have the longest time-series lengths among the rest of the UCR datasets. These 9 datasets are shown in Fig. 5; only one time-series from each class is displayed.

In Fig. 6 raw time-series from the datasets and their representations by APSOS are shown in order to prove that APSOS rep- resentation of each time-series totally follows the original time-series’ shape. According to Fig. 6, except for the time-series with a high frequency, as in Phonomere dataset, the APSOS form has abso- lutely similar shape and characteristics to the original time-series but with a much less number of time steps.
This section of the paper is divided into 3 sub-sections. First, the ISE threshold is estimated for the whole datasets. Second, AP-SOS parameters are tuned by the help of Taguchi method [29]. Finally, the proposed method is compared to the four mentioned methods on 9 datasets.

![Pseudo Code: APSOS Algorithm](image)

---

Table 2

<table>
<thead>
<tr>
<th>Dataset name</th>
<th>Size of train set</th>
<th>Size of test set</th>
<th>Time-series length</th>
</tr>
</thead>
<tbody>
<tr>
<td>CinC_ECG_torso</td>
<td>40</td>
<td>1380</td>
<td>1639</td>
</tr>
<tr>
<td>HandOutlines</td>
<td>370</td>
<td>1000</td>
<td>2709</td>
</tr>
<tr>
<td>Haptics</td>
<td>155</td>
<td>308</td>
<td>1092</td>
</tr>
<tr>
<td>InlineSkate</td>
<td>100</td>
<td>550</td>
<td>1882</td>
</tr>
<tr>
<td>MALLAT</td>
<td>55</td>
<td>2345</td>
<td>1024</td>
</tr>
<tr>
<td>Phoneme</td>
<td>214</td>
<td>1896</td>
<td>1024</td>
</tr>
<tr>
<td>StarLightCurves</td>
<td>100</td>
<td>8236</td>
<td>1024</td>
</tr>
<tr>
<td>UWaveGestureLibrary</td>
<td>896</td>
<td>3582</td>
<td>945</td>
</tr>
</tbody>
</table>

Fig. 4. APSOS algorithm.
Fig. 5. Samples from the 9 datasets.
Fig. 6. Samples from datasets and their representations by APSOS.
Fig. 6. Continued Table 3
Number of samples from each dataset to be analyzed by an expert and averages of the estimations.
5.1. ISE threshold estimation

Most of the methods for time-series dimensionality reduction need the number of segments as an input parameter. Various methods have been proposed in the literature to do so [16] but it’s important to notice that the most accurate way to estimate the required number of segments for a given time-series when there is a shape-based point of view to this problem is by an expert and by analyzing the time-series features by looking at its shape. For example, by looking at the time-series in Fig. 1, the user would understand that 21 is the minimum number of segments required to represent the given time-series. But this is a very time-consuming and boring method and it is useless for the large databases. In this paper, among the considered methods for the comparison, PAA and APCA need only the exact number of segments, but PLA requires not only the exact number of segments but also an initial solution. DSA does not need any input parameters. APSOS does not need the exact number of segments, but a minimum value can help the algorithm find the solution faster. By considering these facts, and to make all the experiment’s conditions equal for all five methods, ISE is used to find the initial solution and number of segments.

In ISE method, each segment is extended until its error exceeds a threshold at a point. Then that point is selected as the endpoint of the current segment and the beginning point of the next segment. The problem here is how to define the amount of threshold. Although according to [16] this threshold depends on the type of data, this section provides experimental results that proves in most of the cases the same amount of threshold can be selected.

In order to prove this fact 10 percent of the data is randomly selected as a sample. As each dataset has different classes and characteristics, the amount of different classes may be different, the 10 percent are sampled from each class rather than each dataset. Since there are totally 19,478 time-series in all 9 datasets, the total number of samples are 1950. This is shown in Table 3. These samples are carefully analyzed and looked at by an expert and the number of segments are estimated for them.

Besides that, the ISE algorithm is applied to every single time-series in all 9 datasets with ISE threshold ranging from 0.01 to 2 by the step of 0.01 (200 amounts for the threshold). It is important to say that each time-series is standardized into [-1, 1] before being tested for the threshold. Data from this experiment is averaged for each dataset which yields to 200 amounts for the number of segments for each dataset. The result for the first dataset can simply be represented in Fig. 7.

Fig. 7 simply shows the number of segments for each threshold old amount, which is averaged specifically from all of the time-series in dataset CinC_ECG_torso. TEST. The rest of the curves are not shown here as they are almost similar to this one. Now this

<table>
<thead>
<tr>
<th>Dataset name</th>
<th>Size of test set</th>
<th>Number of classes</th>
<th>Total number of samples</th>
<th>Average of estimations for the number of segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CinC_ECG_torso</td>
<td>1380</td>
<td>4</td>
<td>140</td>
<td>30</td>
</tr>
<tr>
<td>HandOutlines</td>
<td>10 000</td>
<td>2</td>
<td>101</td>
<td>21</td>
</tr>
<tr>
<td>Haptics</td>
<td>308</td>
<td>5</td>
<td>32</td>
<td>49</td>
</tr>
<tr>
<td>InlineSkate</td>
<td>550</td>
<td>7</td>
<td>59</td>
<td>27</td>
</tr>
<tr>
<td>MALLAT</td>
<td>2345</td>
<td>8</td>
<td>239</td>
<td>71</td>
</tr>
<tr>
<td>Phoneme</td>
<td>1896</td>
<td>39</td>
<td>207</td>
<td>169</td>
</tr>
<tr>
<td>StarLightCurves</td>
<td>8236</td>
<td>3</td>
<td>825</td>
<td>50</td>
</tr>
<tr>
<td>UWaveGestureLibrary</td>
<td>3582</td>
<td>8</td>
<td>361</td>
<td>84</td>
</tr>
<tr>
<td>Worms</td>
<td>181</td>
<td>5</td>
<td>21</td>
<td>85</td>
</tr>
</tbody>
</table>

Table 4

For the ISE threshold for each dataset

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Number of Thresholds</th>
<th>Variance</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>CinC_ECG_torso</td>
<td>140</td>
<td>0.29583</td>
<td>1.25</td>
</tr>
<tr>
<td>HandOutlines</td>
<td>101</td>
<td>8.4156e-30</td>
<td>1.98</td>
</tr>
<tr>
<td>Haptics</td>
<td>32</td>
<td>0.01453</td>
<td>0.18</td>
</tr>
<tr>
<td>InlineSkate</td>
<td>59</td>
<td>0.04477</td>
<td>1.88</td>
</tr>
<tr>
<td>MALLAT</td>
<td>239</td>
<td>0.02398</td>
<td>0.37</td>
</tr>
<tr>
<td>Phoneme</td>
<td>207</td>
<td>0.02627</td>
<td>0.17</td>
</tr>
<tr>
<td>StarLightCurves</td>
<td>825</td>
<td>0.24764</td>
<td>0.32</td>
</tr>
<tr>
<td>UWaveGestureLibrary</td>
<td>361</td>
<td>0.02129</td>
<td>0.13</td>
</tr>
<tr>
<td>Worms</td>
<td>21</td>
<td>0.01937</td>
<td>0.17</td>
</tr>
</tbody>
</table>

results must be integrated with the results of estimating the number of segments for the samples. To do so, for each sample, whose required number of segments is estimated by the expert, the appropriate ISE threshold was found by the help of the curves like the one in Fig. 7. That means, for each dataset, the total number of appropriate ISE thresholds found is equal to the number of samples in the dataset, for example for dataset CinC_ECG_torso_TEST, 140 thresholds were found. For each dataset, these values were averaged and the variances were also calculated which is shown in Table 4.

By averaging the Mean column of Table 4, the universal value for the threshold will be 0.7167. The variances prove that averaging on each row does not make any problems here and is justifiable. Although the variance of the Mean column is 0.59 even averaging on that does not make any problems since increasing or decreasing the threshold for 0.59 will not affect the required number of segments too much. From the information in Table 4 it can be concluded that the threshold range is not too wide.

5.2. Parameter tuning

To tune 9 APSOS parameters including $P,c_1,c_2,w_1,\epsilon_1,\epsilon_2,\delta_1,\delta_2$, the Taguchi method [29] is used. For each parameter 3 levels are considered and therefore 27 experiments with different conditions based on the Taguchi table are conducted, while each experiment is conducted for 10 times and the results are averaged. From each class of each dataset 1 time-series are randomly selected; i.e. in total, 81 time-series are chosen for this experiment. For an example, one of the results is shown in Fig. 8. As a result, 81 values for each parameter are collected as the best and these values are averaged by each class and then by each dataset. The mean value for each parameter and the variance is presented in Table 5.

According to the low variances, this conclusion can be made that the value for each parameter is independent of dataset and can be used universally with any kind of datasets.
Fig. 7. Average number of segments for each ISE threshold in Dataset CinC_ECG_torso_TEST.

Fig. 8. Taguchi results for Dataset 1. The delay 1 and 2 are respectively $\delta_1$ and $\delta_2$.

Table 5
APSOS parameters tuning results.

<table>
<thead>
<tr>
<th>$P$</th>
<th>$\epsilon_1$</th>
<th>$\epsilon_2$</th>
<th>$w$</th>
<th>$v^c$</th>
<th>$\epsilon_1$</th>
<th>$\epsilon_2$</th>
<th>$\delta_1$</th>
<th>$\delta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>100</td>
<td>1.25</td>
<td>1.51</td>
<td>1.56</td>
<td>0.69</td>
<td>0.0085</td>
<td>0.0013</td>
<td>7</td>
</tr>
<tr>
<td>Variance</td>
<td>0</td>
<td>0.068</td>
<td>0.038</td>
<td>0.069</td>
<td>0.018</td>
<td>4.1e-06</td>
<td>1.04e-06</td>
<td>13.25</td>
</tr>
</tbody>
</table>

Table 6
Comparison between APSOS, DSA and other methods in average number of segments (aNS) and compression ratio in the form of percent (C%) for each dataset.

<table>
<thead>
<tr>
<th></th>
<th>CinC_ECG</th>
<th>Hand Outlines Haptics</th>
<th>Inline Skate MALLAT</th>
<th>Phoneme</th>
<th>StarLight Curves UWave</th>
<th>Gesture</th>
<th>Worms</th>
</tr>
</thead>
<tbody>
<tr>
<td>APSOS</td>
<td>73</td>
<td>96</td>
<td>172</td>
<td>94</td>
<td>113</td>
<td>90</td>
<td>102</td>
</tr>
<tr>
<td>DSA</td>
<td>474</td>
<td>71</td>
<td>890</td>
<td>67</td>
<td>334</td>
<td>69</td>
<td>595</td>
</tr>
<tr>
<td>Others</td>
<td>35</td>
<td>98</td>
<td>88</td>
<td>97</td>
<td>29</td>
<td>97</td>
<td>46</td>
</tr>
</tbody>
</table>
Fig. 9. Comparison results for each dataset.
Fig. 9. Continued
Fig. 9. Continued
5.3. Comparison of methods

In this section, the selected methods of time-series segmentation including PLA, PAA, APCA and DSA are all compared to the proposed algorithm on the mentioned datasets. The measure of comparison among segmentation methods is Reconstruction Error or MSE. Although here, techniques are being evaluated by the amount of Reconstruction Error, the number of segments is also considered as a side measure of the comparison. From each dataset whose number of time-series is greater than 1000, one thousands time-series are selected randomly and for the rest of them the whole dataset are selected. Each time-series are standardized into...
6. Conclusion

In this paper a new perspective of time-series dimensionality reduction specifically segmentation was proposed with an ap-proach of optimization. The proposed algorithm for time-series segmentation, APSOS, has many superiorities to other methods for this application and the experimental results show that APSOS out-performs other methods in the area of segmentation. First, contrary to most of the techniques in this area, it doesn’t need any input parameters since all of the parameters were tuned and uni-versal values compatible to most of the datasets were presented in this paper. APSOS is definitely adaptive to each time-series and develops the solution in an intelligent way that saves most of the characteristics and features of the given time-series. Second, the compression ratio for APSOS technique is very high (about 90%) in most of the cases); this makes APSOS appropriate for very long time-series as it makes them short enough for further analysis. Third, not only does APSOS provide solution with equal or less time steps than other methods, but also its Reconstruction Error is less than them. Again this proves that APSOS’s solutions for seg-mentation, inherit the original time-series shape-based character-istics. Finally, many distance measures for time-series are shape-based and work properly only on the basis of the time-series shape and shape-based characteristics, and as APSOS preserves the time-series shape, its solutions can easily be used with shape-based dis-tance measures specifically those which are based on time-warping such as Dynamic Time Warping.

References


Table 7
Average execution time of each algorithm for each dataset (in seconds).

<table>
<thead>
<tr>
<th></th>
<th>Conc. ECG</th>
<th>Hand Outlines</th>
<th>Haptics</th>
<th>Inline Skate</th>
<th>MALLAT</th>
<th>Phoneme</th>
<th>StarLight Curves</th>
<th>UWave Gesture</th>
<th>Worms</th>
</tr>
</thead>
<tbody>
<tr>
<td>APSOS</td>
<td>7.62</td>
<td>11.87</td>
<td>5.41</td>
<td>9.31</td>
<td>6.53</td>
<td>3.21</td>
<td>10.59</td>
<td>5.42</td>
<td>4.77</td>
</tr>
<tr>
<td>DSA</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>PAA</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>PLA</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>APCa</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
<td>&lt; 1</td>
</tr>
</tbody>
</table>


