Focus Article

Data representation for time series data mining: time domain approaches

Seunghye J. Wilson*

In most time series data mining, alternate forms of data representation or data preprocessing is required because of the unique characteristics of time series, such as high dimension (the number of data points), presence of random noise, and nonlinear relationship of the data elements. Therefore, any data representation method aims to achieve substantial data reduction to a manageable size, while preserving important characteristics of the original data, and robustness to random noise. Moreover, appropriate choice of a data representation method may result in meaningful data mining. Many high level representation methods of time series data are based on time domain approaches. These methods preprocess the original data in the time domain directly and are useful to understand the behavior of data over time. Piecewise approximation, data representation by identification important points, and symbolic representation are some of the main ideas of time domain approaches, and widely used in various fields. © 2016 Wiley Periodicals, Inc.

How to cite this article:
WIREs Comput Stat 2017, 9:e1392. doi: 10.1002/wics.1392

Keywords: time series data mining, data preprocessing, data reduction, high-level data representation, time domain approaches

INTRODUCTION

Time series is an important form of data in various domains of industry and research, and the interest in time series data mining has exploded in recent decades. However, it is awkward to apply mining techniques to the observed data directly because of the unique characteristics of time series: large amount of data, presence of random noise, and nonlinear relationships of data elements. As a result, data representation in the simplified form, or data preprocessing is an essential step in time series data mining. The main purpose of data representation is to reduce data to a manageable size or to approximate the data by removing random noise. The reduced data, however, must preserve the important local and global features of the original data.

Time domain approaches for data representation are especially useful for understanding the behavior of the data over time. They summarize the original data by approximating interval values, identifying critical points, or converting numerical data to discrete variables. Piecewise approximation is one of the most common time domain approaches. These methods represent the original data based on non-overlapping intervals over time. The resulting data by piecewise approximation may be a sequence of continuous or discontinuous straight lines, or of representative values of all intervals with substantially reduced length. Another common approach for data representation is to preserve critical points that contribute to revealing important features, such as the overall shape or trend change points, of the original data. Recently, as interest in mining of

*Correspondence to: swilso16@masonlive.gmu.edu

Department of Statistics, George Mason University, Fairfax, VA, USA

Conflict of interest: The author has declared no conflicts of interest for this article.

massive sized data, so-called 'big data,' continues to rise, data representation methods by transforming numeric time series to discrete variable, or symbols, become more and more popular. Symbolic transformation enables not only data reduction but also efficient computation and usage of memory space for data storage since the fewer bits are required for string data than for numeric data. In this article, we review three types of popular approaches for data representation in time domain and discuss their properties.

GENERAL APPROACHES FOR DATA APPROXIMATION

Global Models vs. Piecewise Approximation

In data analysis, global models are commonly used to find simpler representations when the known underlying model is too complicated, or estimate an unknown function given observed data available. These global models are useful to understand data generating processes. For example, linear regression models the linear relationship between explanatory (independent) and response (dependent) variables based on some assumptions such that the variance of the error terms is constant and independent. Polynomial regression is an extension of linear regression models which allows n-th degree polynomial explanatory variables in linear model. ARMA model, particularly for time series data, describes a stochastic process in the forms of polynomials of autoregression and moving average. These models usually depend on specific assumptions and are adequate up to some amount of data points, but unlikely to be tenable as the data size increases.

When the data size is very large, piecewise approximation, such as with piecewise polynomials and splines, is often more effective. Indeed, many time series representation methods are based on piecewise approximation because time series data is often characterized by large size and the presence of random noise. By piecewise model approximation, all data points are divided into a number of non-overlapping segments to build a local model $\mu_i(t)$ ($b_{i-1} \le t < b_i$, $b_0 = t_1$), in each segment, and the original data is represented by a sequence of local models $\{\mu_1(t), \cdots, \mu_i(t), \cdots, \mu_n(t)\}$. Hence, given time series $X = x_1, \cdots, x_N$ the model is written as

$$\hat{X}(t) = \sum_{i=0}^{n} \mu_i(t), \; n \ll N \tag{1}$$

Batch and Online Processing

Large size data can be approximated or represented by batch processing or online processing based on its availability when analyzed. Batch processing is used when all data points are available during the computation, and once data processing begins, acquisition of new data points cannot occur. Hence, it is necessary to understand data structure in advance of data analysis. On the other hand, online processing analyzes data upon receiving data points continuously and acquiring new data points during the computation. Therefore, the results of data processing are obtained immediately or in a short time and less data storage is required. For this reason, online processing is generally preferred in massive streaming data mining.

TIME SERIES DATA REPRESENTATION

Piecewise Approximation

One simple and common approach for data representation is *piecewise approximation*. Generally, the algorithms of piecewise approximation segment the entire dataset into some number of non-overlapping intervals over time, and fit local models in the intervals. Formally, given time series $X = \{x_t | t = 1, 2, \cdots, N\}$, where t is time index, the entire dataset is divided into k disjoint subsets ($k \ll N$) such that

$$\begin{aligned} X_1 &= \{x_t | t = 1, \cdots, b_1\} \\ X_2 &= \{x_t | t = b_1 + 1, \cdots, b_2\} \\ &\vdots \\ X_k &= \{x_t | t = b_{k-1} + 1, \cdots, N\} \end{aligned}, \tag{2}$$

where $b_1, \cdots, b_{k-1}$ ($b_i < b_{i+1}$, forall i) are breakpoints, and $X_1 \cup \cdots \cup X_k = X$. In piecewise approximation, segmenting data over time and determining the local model are the main objectives. The length of segments or the number of the segments (k in Eq. (2)) might be determined by a predefined constant number over time. Or, the length of each segment may be determined based on the homogeneity of some properties for aggregate data, e.g., small variations or similar trends. In the latter case, the length of segments are often determined by identification of breakpoints where some property of interest of the local model significantly changes, thereby this method may focus on identifying critical points such as points at which the trend changes, while piecewise approximation with constant length for all segments can be more helpful to understand the overall trend of data over time.

The form of local model for segments may be determined by some representative value or by a parametric model. One simple local model is mean value. Using the mean value, the original data is represented as *piecewise constant functions* or *step functions*. Linear line and polynomial models also might be used along with the trend of piecewise aggregate data. Instead of using mean, sum of variation[1] or volatility might be used as a representative value of the data points in each segment. The choice of segmenting and the form of local model, of course, must be made considering the purpose of analysis and mining.

EXAMPLES: Piecewise Approximation

Piecewise Aggregate Approximation

Piecewise aggregate approximation (PAA[2,3]), or piecewise constant approximation, is simple to use and performs very well on indexing. Indexing is a time series mining task that find the most similar time series in a database given a query time series and a similarity measures. First, the original data is normalized, and then the normalized data is divided into equal length and non-overlapping intervals. Finally, the reduced data is represented by the mean-value of data points in all segments. Specifically, a normalized time series $C = \{c_1, c_2, \cdots, c_N\}$ is represented as $\overline{C} = \{\overline{c}_1, \overline{c}_2, \cdots, \overline{c}_m\}\ (1 \le m \le N)$, where $\overline{c}_i$ is the mean value of the i-th segment,

$$\overline{c}_i = \frac{m}{N} \sum_{j=\frac{N}{m}(i-1)+1}^{\frac{N}{m}i} c_j. \qquad (3)$$

These m equal length segments, called *frames*, are converted to mean values of data within the frame, and the vector of these mean values is the reduced representation of C. Hence, the represented data is the same as the original data when $m = N$, and the mean value of original data when $m = 1$. The number of segments m can be user-specified parameter, and thus it is flexible to adjust the resolution of the reduced data. In Eq. (3), we assume that m is a factor of N. For the case that m is not a factor of N, that is the length of a given time series is greater or less than N, see Keogh *et al.*[2] and Chakrabarti and Mehrotra.[4]

Adaptive Piecewise Constant Approximation

Adaptive piecewise constant approximation (APCA[5]) has been proposed to relax the equal-length segments assumption of PAA. It still uses the mean values of the segments while allowing them to have different lengths. As a result, APCA can segment the original data *better* with smaller reconstruction errors than PAA. To reduce the reconstruction error, APCA tends to have more breakpoints in a segment of highly volatile data. On the other hand, there are fewer breakpoints in a segment of low volatility data. First, the breakpoints are identified by Haar wavelet transformation, that is the optimal solution of a wavelet compression. Then, the solutions are converted back to their time domain representation. Thus, the reduced data $\overline{C}$ of the original time series $C = \{c_1, c_2, \cdots, c_N\}$ contains the mean values of the data in segments, and the lengths of the segments by recording the right breakpoints of all segments as follows.

$$\overline{C} = \{(cv_1, cr_1), \ldots, (cv_n, cr_n)\}, \quad cr_0 = 0, \quad (n \ll N) \qquad (4)$$

where cv_i is the mean value of data in the i-th segment, and cr_i is the right endpoint of the i-th segment. The length of the i-th segment can be obtained by $cr_i - cr_{i-1}$, $i = 1, \cdots, n$.

Segmented Sum of Variation Features

In time series data mining, many similarity measures proposed are based on Euclidean distance. Often, standardization of data is required before applying similarity measure between time series data since Euclidean distance is sensitive to the noise and the vertical scale of data. Lee *et al.*[1] proposed *segmented sum of variation* (SSV). This method is developed based on the idea that the sum of variation is invariant under the vertical shifting of data. First, time series datasets for comparison are divided into the n segments with equal length, and then the sum of variation for all segments is calculated. Specifically, the algorithm creates n segments ($n << N$) of s points each from the original time series $C = \{c_1, \ldots, c_N\}$, overlapping by sharing a point at the boundary between two adjacent segments.

$$(c_{1,1}, \cdots, c_{1,s}), (c_{2,1}, \cdots, c_{2,s}), \cdots, (c_{n,1}, \cdots, c_{n,s}), \qquad (5)$$

where $c_{i,s} = c_{i+1,1}$ $(i = 1, \cdots, n-1)$. Note that the breakpoints are shared by two adjacent segments. That is, the end points of the i-th segment also become the starting point of the $(i+1)$-th points $(i = 1, \cdots, n-1)$. The sum of variation of the i-th segment is given by

$$\sum_{j=1}^{s-1} |c_{i,j} - c_{i,j+1}|. \qquad (6)$$

Therefore, the reduced data is represented as a sequence of the sum of variation for segments with length n.

Identification of Important Points

While piecewise approximation represents data by fitting local models or capturing some statistics of segments, data representation by identification of important points focuses on selecting a subset of points from the entire dataset. These selected data points contribute critically to the feature of the original data. Although the 'importance' of the points can be defined depending on the feature that the user would like to find from the data, many approaches for data reduction in the time domain attempt to find points that contribute to the *shape* of the original data, e.g., when a sudden jump or drop occurs. If all data points are available before preprocessing, we can analyze the overall data structure and choose the important points continuously for the entire dataset by the criteria of importance (batch processing). Otherwise, we may apply this criteria to a group of data points sequentially as new data is updated to identify critical points (online processing). The following two examples are data representation methods by identifying important points by batch and online processing.

EXAMPLES: Identification of Important Points

Perceptually Important Points[6]

Some data points in time series data may be more influential on the shape of the data while others may be ignored as noise. Patterns used in technical analysis for financial markets are usually identified based on these influential points such as local minimum or maximum. Chung *et al.*[6] proposed *perceptually important points* (PIPs) which are most influential points on the shape of the data for data reduction. These PIPs are selected sequentially based on the perpendicular or vertical distance from the straight line between two previous important points. Specifically, given time series $x_1, x_2, \cdots, x_n$, the first and the last points x_1 and x_n are always the first and the second PIPs, P_1 and P_2. Then, the third PIP, P_3 is identified based on the perpendicular or vertical distance from the straight line between P_1 and P_2. That is, the points at the maximum distance from $\overline{P_1P_2}$ is P_3. The points in the maximum distance from $\overline{P_1P_3}$ and $\overline{P_2P_3}$ is identified as the fourth PIP, P_4. In the same way, to find the k-th PIP, P_k, the algorithm searches for the point in the maximum distance from $k - 2$ straight lines between adjacent PIPs until it identifies a predefined number of PIPs. This approach is obviously batch processing since all data points are required at the time of analysis to identify the first and the second PIPs, x_1 and x_n.

Compression by Extracting Major Extrema[7]

With the idea that local minima and maxima can be good candidates for important points that influence on the shape of data, Fink and Gandhi[7] proposed compression by investigating extrema (minima and maxima). Among all minima and maxima, the algorithm chooses important points that contribute to creating some magnitude of fluctuation or larger, and discards the rest of data points. The "importance" of extrema is defined by a threshold parameter $R > 0$, which is a minimal magnitude of "important" fluctuation. For example, given a time series $x_i, \cdots, x_j$ and a positive value R, x_k $(i < k < j)$ is an *important* minimum (maximum) if (1) $x_k = \min\{x_i, \cdots, x_j\}$ $(x_k = \max\{x_i, \cdots, x_j\})$, and (2) $dist(x_k, x_i) \geq R$ and $dist(x_k, x_j) \geq R$, where $dist(a, b)$ is a distance between a and b such that $|a-b|, \frac{|a-b|}{|a|+|b|}$, or $\frac{|a-b|}{max(|a|,|b|)}$. Thus, a large value of R implies a *high compression rate*; that is, the selection of a few number of extrema. This algorithm can be used not only for batch processing but also online processing for fast indexing.

Symbolic Data Representations

Another common approach for time series representation is to convert numeric data into a finite number of discrete variables, generally symbolic variables. Converting numerical values into strings saves memory space and enables fast computation. One simple method of symbolic representation is to map numerical data within a certain range of values onto a symbol. Given a time series $X = \{x_i | x_i \in \mathcal{R}, i = 1, \cdots, N\}$, it is mapped to symbolic series $S = \{s_i | s_i \in \mathcal{C}, i = 1, \ldots, N\}$ by symbolic representation, where $\mathcal{C}$ is a collection of symbols. Another popular method is to discretize piecewise data and then convert these piecewise data into strings. That is, the data representation consists of two steps: first, piecewise approximation and then, transformation of the reduced data obtained from the first step into symbols. The latter method enables data reduction as well as memory space saving and more efficient computation while the dimension of the original data does not change by the former method. The next two examples describe details about symbolic data representation.

EXAMPLES: Symbolic Representations

Shape Description Alphabet[8]

Shape description alphabet (SDA) is proposed for blurry searching in a large time series database. This method transforms the difference between two

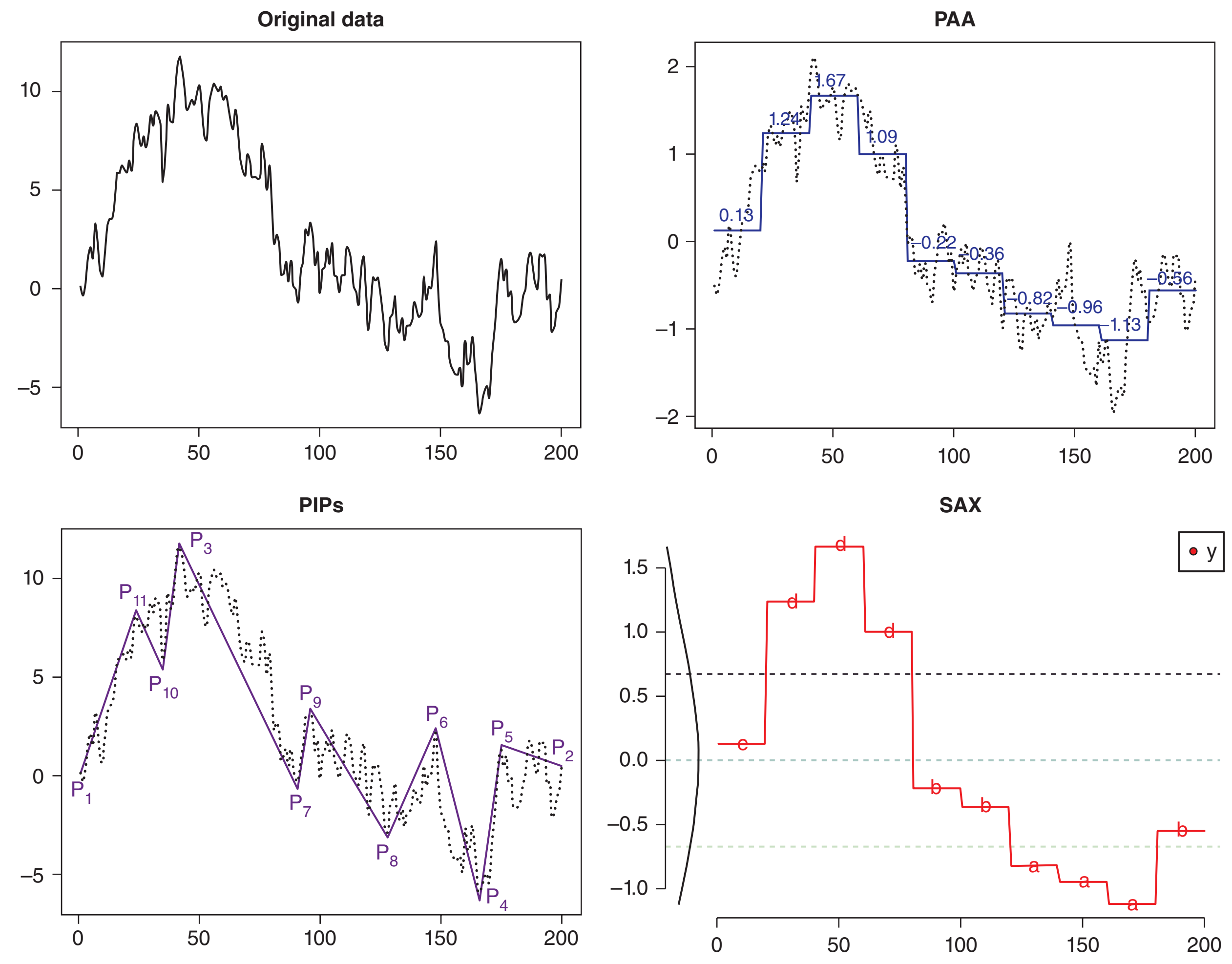

FIGURE 1 | Time series representation by *piecewise aggregate approximation* (PAA), *perceptually important points* (PIPs), and *symbolic aggregate approximation* (SAX). The dimension of the original data has been reduced from $N = 200$ to $n = 10$ by PAA and SAX, and to $n = 11$ by PIPs.

adjacent points, x_i and x_{i+1}, that is $d_i = x_{i+1} - x_i$, to a finite set of letters. For example, it uses **a**, **u**, **s**, **d**, and **e** for highly increasing, slightly increasing, stable, slightly decreasing, and highly decreasing transactions, respectively. The cutoff points, *lvalue* (lower bound) and *hvalue* (upper bound), to determine a symbolic value for each d_i is obtained based on the distribution of d_i. Thus, the prior knowledge about d_i's is necessary to find optimal cutoff points. SDA is not appropriate for noisy data because the difference di is highly influenced by random error and results in failure to capture the general shape of the original data.[9]

Symbolic Aggregate Approximation

Symbolic aggregate approximation (SAX)[10] represents time series data by two steps. First, SAX uses normalized data to represent by PAA, and then the coefficients obtained from PAA are converted into alphabet strings. Thus, two parameters are required for SAX representation: the number of symbols (*alphabet size*) and the dimension of the reduced data (the length of the reduced data). Given time series $C = \{c_1, \cdots, c_N\}$, coefficients of the reduced data $\overline{C} = \{\overline{c}_1, \ldots, \overline{c}_n\} (n \ll N)$ by PAA are ##converted based on the quantiles values of $\overline{c}_i$'s. Specifically, with predefined symbol set $\{L_1, \cdots, L_a\}$ (*symbol size* = a), SAX finds the breakpoints $\{\beta_1, \cdots, \beta_{a-1}\}$ to determine symbolic values such that $P(Z < \beta_1) = P(\beta_1 \le Z < \beta_2) = \cdots = P(\beta_{a-1} \le Z)$, where $Z \sim N(0,1)$. Then, each coefficient $\overline{c}_i$ in PAA approximation is transformed to a symbol $\hat{c}_i$ by

$$\hat{c}_i = L_j \text{ if and only if } \overline{c}_i \in [\beta_{j-1}, \beta_j), \tag{7}$$

where $i = 1, \cdots, n$ and $j = 1, \cdots, a$. SAX is widely used in time series data mining due to its advantages of fast computing and substantial dimensionality reduction.

CONCLUSIONS

The ultimate goal of data representation is dimemsionality reduction and important feature extraction from the data to enable meaningful mining tasks, such as classification, clustering, indexing, etc. These two properties, data reduction and feature extraction, are shown in all data representation approaches. Although there are a lot of methods that have been suggested, there is no single superior method over all others. Instead, the features that the user would like to find from data should be considered to choose an appropriate data representation method. Figure 1 illustrates time series representations by three different methods.

Streaming data representation is challenging because of its tremendous size and incoming speed, but this area is obviously promising as the interest in 'big data' continues to rise recently. Furthermore, selecting an appropriate similarity measure is essential for data mining along with data representation. Due to the unique properties of time series data, large size and noisy, similarity measures commonly used, e.g., L_p norms are not likely feasible to measure two time series data, thus most time series representation methods are generally proposed with similarity measures. Therefore, the applicability of similarity measure to the reduced data is also an important consideration in data representation.

FURTHER READING

Fu T. A review on time series data mining. *Eng Appl Artif Intell* 2011, 24:164–181.

Ratanamahatana C, Lin J, Gunopulos D, Keogh E, Vlachos M, Das G. Mining time series data. In: *Data Mining and Knowledge Discovery Handbook*. New York: Springer; 2009, 1049–1077.

Philippe E, Agon C. Time-series data mining. *ACM Comput Surv* 2012, 45:1–34.

REFERENCES

1. Lee S, Kwon D, Lee S. Dimensionality reduction for indexing time series based on the minimum distance. *J Inf Sci Eng* 2003, 19:697–711.
2. Keogh E, Chakrabarti K, Pazzani M, Mehrotra S. Dimensionality reduction for fast similarity search in large time series databases. In: *Knowledge and Information Systems*, vol. 3. New York: Springer; 2001, 263–286.
3. Yi B, Faloutsos C. Fast time sequence indexing for arbitrary norms. In: *Proceedings of the 26th International Conference on Very Large Data Bases*. San Francisco: Morgan Kaufmann Publishers Inc; 2000, VLDB '00: 385–394.
4. Chakrabarti K, Mehrotra S. The hybrid tree: an index structure for high dimensional feature spaces. In: *Proceedings of 15th International Conference on Data Engineering*, IEEE; 1999, 440–447.
5. Keogh E, Chakrabarti K, Pazzani M, Mehrotra S. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record* 2001, 30:151–162.
6. Chung F, Fu T, Luk R, Ng V. Flexible time series pattern matching based on perceptually important points. In: *International Joint Conference on Artificial Intelligence Workshop on Learning from Temporal and Spatial Data*, 2001, 1–7.
7. Fink E, Gandhi H. Compression of time series by extracting major extrema. *J Exp Theor Artif Intell* 2011, 23:255–270.
8. André-Jönsson H, Dushan ZB. *Using Signature Files for Querying Time-Series Data*. New York: Springer; 1977, 211–220.
9. Lin J, Keogh E, Wei L, Lonardi S. Experiencing SAX: a novel symbolic representation of time series. In: *Data Mining and Knowledge Discovery*, vol. 15. New York: Springer; 2007, 107–144.
10. Lin J, Keogh E, Wei L, 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*, ACM, 2003.