Chapter 3

Similarity Measures and Dimensionality Reduction Techniques for Time Series Data Mining

Carmelo Cassisi, Placido Montalto, Marco Aliotta, Andrea Cannata and Alfredo Pulvirenti

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/49941

1. Introduction

A time series is *"a sequence* $X = (x_1, x_2, \ldots, x_m)$ *of observed data over time"*, where m is the number of observations. Tracking the behavior of a specific phenomenon/data in time can produce important information. A large variety of real world applications, such as meteorology, geophysics and astrophysics, collect observations that can be represented as time series.

A collection of time series can be defined as a Time Series Database (*TSDB*). Given a *TSDB*, most of time series mining efforts are made for the similarity matching problem. Time series data mining can be exploited from research areas dealing with signals, such as image processing. For example, image data can be converted to time series: from image color histograms (Fig. 2), where image matching can be applied, to object perimeters for the characterization of shapes [39].

Time series are essentially high-dimensional data [17]. Mining high-dimensional involves addressing a range of challenges, among them: i) the curse of dimensionality [1], and ii) the meaningfulness of the similarity measure in the high-dimensional space. An important task to enhance performances on time series is the reduction of their dimensionality, that must preserve the main characteristics, and reflects the original (dis)similarity of such data (this effect will be referred to as *lower bounding* [11]). When treating time series, the similarity between two sequences of the same length can be calculated by summing the ordered point-to-point distance between them (Fig. 3). In this sense, the most used distance function is the *Euclidean Distance* [13], corresponding to the second degree of general L_p*-norm* [41]. This distance measure is cataloged as a metric distance function, since it obeys to the three

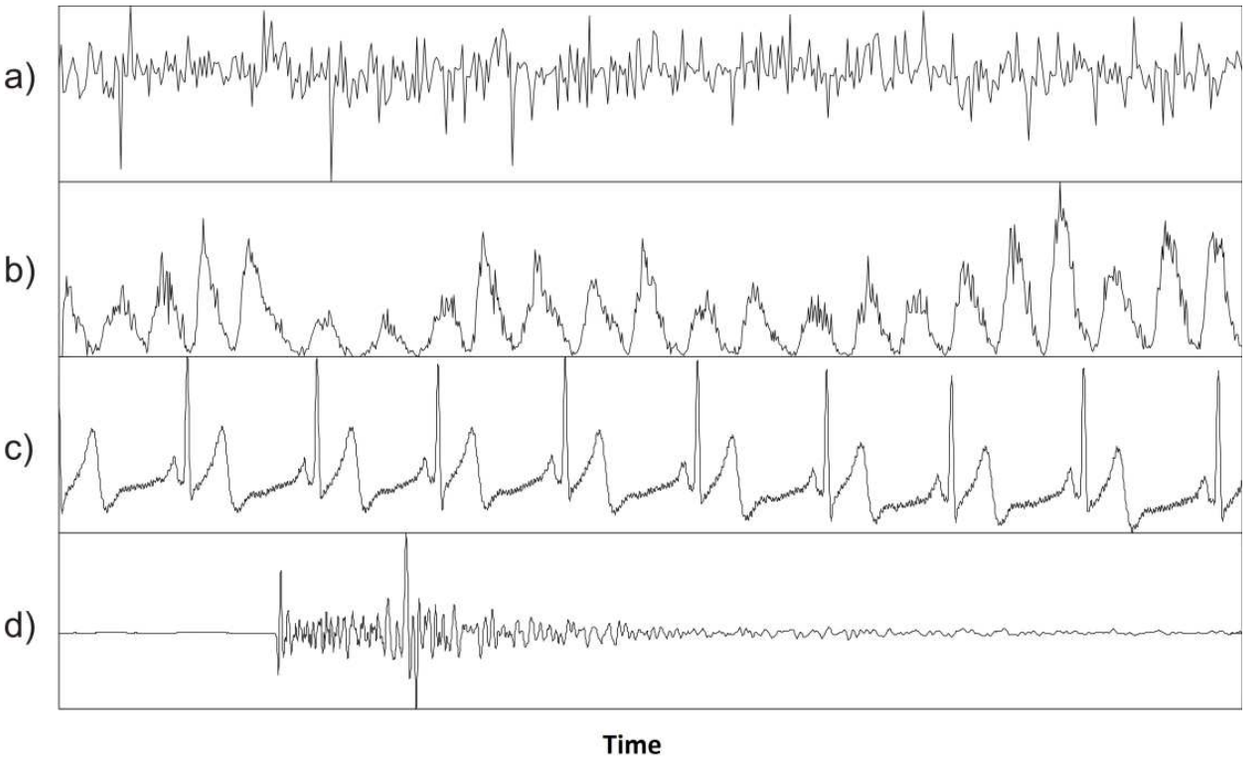

Figure 1. Examples of time series data relative to a) monsoon, b) sunspots, c) ECG (ElectroCardioGram), d) seismic signal.

fundamentals metric properties: *non-negativity*, *symmetry* and *triangle inequality* [29]. In most cases, a metric function is desired, because the triangle inequality can then be used to prune the index during search, allowing speed-up execution for exact matching [28]. In every way, Euclidean distance and its variants present several drawbacks, that make inappropriate their use in certain applications. For these reasons, other distance measure techniques were proposed to give more robustness to the similarity computation. In this sense it is required to cite also the well known *Dynamic Time Warping* (*DTW*) [22], that makes distance comparisons less sensitive to signal transformations as shifting, uniform amplitude scaling or uniform time scaling. In the literature there exist other distance measures that overcome signal transformation problems, such as the *Landmarks similarity*, which does not follow traditional similarity models that rely on point-wise Euclidean distance [31] but, in correspondence of human intuition and episodic memory, relies on similarity of those points (times, events) of "greatest importance" (for example local maxima, local minima, inflection points).

In conjunction to this branch of research, a wide range of techniques for dimensionality reduction was proposed. Among them we will treat only representations that have the desirable property of allowing *lower bounding*. By this property, after establishing a true distance measure for the raw data (in this case the Euclidean distance), the distance between two time series, in the reduced space, results always less or equal than the true distance. Such a property ensures exact indexing of data (i.e. with no false negatives [13]). The following representations describe the state-of-art in this field: spectral decomposition through *Discrete Fourier Transform* (*DFT*) [1]; *Discrete Wavelet Transform* (*DWT*) [7]; *Singular*

Value Decomposition (*SVD*) [24]; *Piecewise Aggregate Approximation* (*PAA*) [19]; *Adaptive Piecewise Constant Approximation* (*APCA*) [6]; *Piecewise Linear Approximation* (*PLA*) [20]; and *Chebyshev Polynomials* (*CHEB*) [29]. Many researchers have also included symbolic representations of time series, that transform time series measurements into a collection of discretized symbols; among them we cite the *Symbolic Aggregate approXimation* (*SAX*) [26], based on *PAA*, and the evolved multi-resolution representation *iSAX 2.0* [35]. Symbolic representation can take advantage of efforts conducted by the text-processing and bioinformatics communities, who made available several data structures and algorithms for efficient pattern discovery on symbolic encodings [2],[25],[36].

The chapter is organized as follows. Section 2 will introduce the similarity matching problem on time series. We will note the importance of the use of efficient data structures to perform search, and the choice of an adequate distance measure. Section 3 will show some of the most used distance measure for time series data mining. Section 4 will review the above mentioned dimensionality reduction techniques.

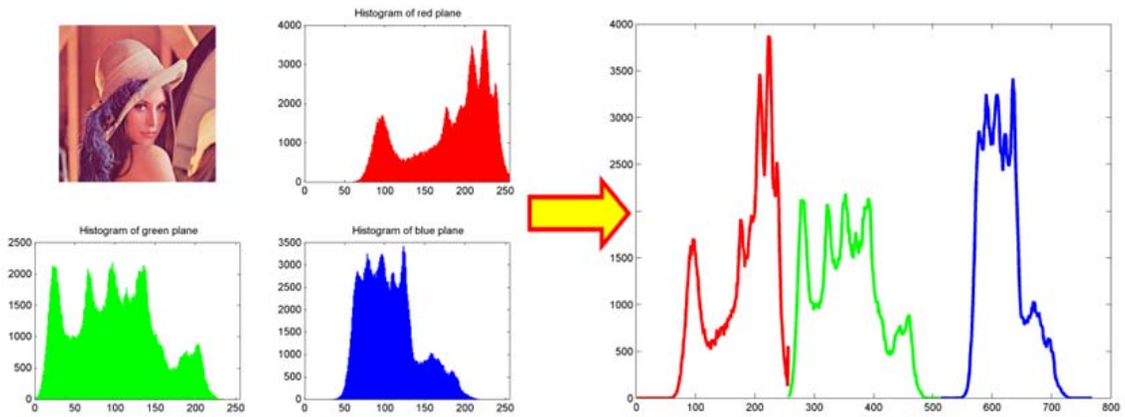

Figure 2. An example of conversion from the RGB (Red, Green, Blue) image color histograms, to a time series.

2. Similarity matching problem

A *TSDB* with m objects, each of length n, is denoted by $TSDB = \{x_1, x_2, \ldots, x_m\}$, where $x_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)})$ is a vector denoting the ith time series and $x_i^{(j)}$ denotes the jth values of x_i, with respect to time. n indicates the *dimensionality* of the data set.

The general representation of a *TSDB* is a $m \times n$ matrix:

$$A = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(j)} & \cdots & x_1^{(n)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(j)} & \cdots & x_2^{(n)} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_i^{(1)} & x_i^{(2)} & \cdots & x_i^{(j)} & \cdots & x_i^{(n)} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ x_m^{(1)} & x_m^{(2)} & \cdots & x_m^{(j)} & \cdots & x_m^{(n)} \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \quad (1)$$

In many cases, datasets are supported by special data structures, especially when dataset get larger, that are referred as *indexing* structures. Indexing consists of building a data structure I that enables efficient searching within database [29]. Usually, it is designed to face two principal similarity queries: the (i) *k-nearest neighbors* (*knn*), and the (ii) *range query* problem. Given a time series Q in *TSDB*, and a similarity/dissimilarity measure $d(T,S)$ defined for each pair T, S in *TSDB*, the former query deals with the search of the set of first k time series in *TSDB* more similar to Q. The latter query finds the set R of time series that are within distance r from Q. Given an indexing structure I, there are two ways to post a similarity query in time series databases [29]:

- *whole matching*: given a *TSDB* of time series, each of length n, whole matching relates to computation of similarity matching among time series along their whole length.
- *subsequence matching*: given a *TSDB* of m time series $S_1, S_2, \ldots, S_m$, each of length n_i, and a short query time series Q of length $n_q < n_i$, with $0 < i < m$, subsequence matching relates to finding matches of Q into subsequences of every S_i, starting at every position.

Indexing is crucial for reaching efficiency on data mining tasks, such as *clustering* or *classification*, specially for huge database such as *TSDB*s. Clustering is related to the unsupervised division of data into groups (clusters) of similar objects under some similarity or dissimilarity measures. Sometimes, on time series domain, a similar problem to clustering is the *motif discovery* problem [28], consisting of searching main cluster (or motif) into a *TSDB*. The search for clusters is unsupervised. Classification assigns unlabeled time series to predefined classes after a supervised learning process. Both tasks make massive use of distance computations.

Distance measures play an important role for similarity problem, in data mining tasks. Concerning a distance measure, it is important to understand if it can be considered *metric*. A metric function on a *TSDB* is a function $f : TSDB \times TSDB \rightarrow \mathrm{R}$ (where R is the set of real numbers). For all x, y, z in *TSDB*, this function obeys to four fundamental properties:

$$f(x, y) \geq 0 \quad (non\text{-}negativity) \tag{2}$$

$$f(x, y) = 0 \text{ if and only if } x = y \quad (identity) \tag{3}$$

$$f(x, y) = f(y, x) \quad (symmetry) \tag{4}$$

$$f(x, z) \leq f(x, y) + f(y, z) \quad (triangle\ inequality) \tag{5}$$

If any of these is not obeyed, then the distance is non-metric. Using a metric function is desired, because the triangle inequality property (Eq. 5) can be used to index the space for speed-up search. A well known framework, for indexing time series, is *GEMINI* (*GEneric Multimedia INdexIng*) [13] that designs fast search algorithms for locating time series that match, in an exact or approximate way, a query time series Q.

It was introduced to accommodate any dimensionality reduction method for time series, and then allows indexing on new representation [29]. *GEMINI* guarantees no false negatives on index search if two conditions are satisfied: (i) for the raw time series, a metric distance

measure must be established; (ii) to work with the reduced representation, a specific requirement is that it guarantees the lower bounding property.

3. Similarity measures

A common data mining task is the estimation of similarity among objects. A similarity measure is a relation between a pair of objects and a scalar number. Common intervals used to mapping the similarity are [-1, 1] or [0, 1], where 1 indicates the maximum of similarity.

Considering the similarity between two numbers x and y as :

$$numSim(x,y) = 1 - \frac{|x-y|}{|x|+|y|} \tag{6}$$

Let two time series $X = x_1,\ldots,x_n$, $Y = y_1,\ldots,y_n$, some similarity measures are:

- mean similarity defined as:

$$tsim(X,Y) = \frac{1}{n}\sum_{i=1}^{n} numSim(x_i,y_i) \tag{7}$$

- root mean square similarity:

$$rtsim(X,Y) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} numSim(x_i,y_i)^2} \tag{8}$$

- and peak similarity [15]:

$$psim(X,Y) = \frac{1}{n}\sum_{i=1}^{n}\left[1 - \frac{|x_i-y_i|}{2\max\left(|x_i|,|y_i|\right)}\right] \tag{9}$$

Another common similarity functions used to perform complete or partial matching between time series are the *cross-correlation* function (or *Pearson's correlation* function) [38] and the cosine angle between the two vectors. The cross correlation between two time series x and y of length n, allowing a shifted comparison of l positions, is defined as:

$$r_{XY} = \frac{\sum_{i=1}^{n}\left(x_i-\overline{X}\right)\left(y_{i-l}-\overline{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_i-\overline{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(y_{i-l}-\overline{Y}\right)^2}} \tag{10}$$

Where $\overline{X}$ and $\overline{Y}$ are the means of X and Y. The correlation r_{XY} provides the degree of linear dependence between the two vectors X and Y from perfect linear relationship (r_{XY} = 1), to perfect negative linear relation (r_{XY} = -1).

Another way to evaluate similarity between two vectors is the estimation of cosine of the angle between the two vectors X and Y, defined as:

$$\cos(\vartheta) = \frac{X \cdot Y}{\|X\|\|Y\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{11}$$

This measure provides values in range [-1, 1]. The lower boundary indicates that the X and Y vectors are exactly opposite, the upper boundary indicates that the vectors are exactly the same, finally the 0 value indicates the independence.

3.1. Metric distances

Another way to compare time series data involves concept of distance measures. Let be two time series T and S vectors of length n, and T_i and S_i the ith values of T and S, respectively. Let us list the following distance measures. This subsection presents a list of distance functions used in Euclidean space.

1. **Euclidean Distance**. The most used distance function in many applications. It is defined as (Fig. 3):

$$d(T,S) = \sqrt{\sum_{i=1}^{n} (T_i - S_i)^2} \tag{12}$$

2. **Manhattan Distance**. Also called "city block distance". It is defined as:

$$d(T,S) = \sum_{i=1}^{n} |T_i - S_i| \tag{13}$$

3. **Maximum Distance**. It is defined to be the maximum value of the distances of the attributes:

$$d(T,S) = \max_{0<i\leq n} |T_i - S_i| \tag{14}$$

4. **Minkowski Distance**. The Euclidean distance, Manhattan distance, and Maximum distance, are particular instances of the Minkowski distance, called also L_p-norm. It is defined as:

$$d(T,S) = \sqrt[p]{\sum_{i=1}^{n} (T_i - S_i)^p} \tag{15}$$

where p is called the order of *Minkowski* distance. In fact, for Manhattan distance $p = 1$, for the Euclidean distance $p = 2$, while for the Maximum distance $p = \infty$.

5. **Mahalanobis Distance**. The Mahalanobis distance is defined as:

$$d(T,S) = \sqrt{(T-S)W^{-1}(T-S)^T} \tag{16}$$

where W is the *covariance* matrix [12].

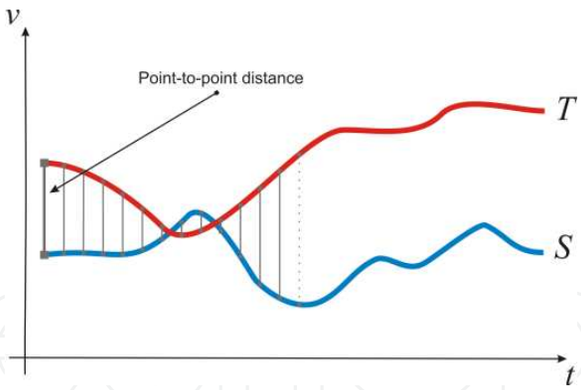

Figure 3. *T* and *S* are two time series of a particular variable *v*, along the time axis *t*. The Euclidean distance results in the sum of the point-to-point distances (gray lines), along all the time series.

3.2. Dynamic time warping

Euclidean distance is cataloged as a metric distance function, since it obeys to the metric properties: *non-negativity*, *identity*, *symmetry* and *triangle inequality* (Section 2, Eq.2~5). It is surprisingly competitive with other more complex approaches, especially when dataset size gets larger [35]. In every way, Euclidean distance and its variants present several drawbacks, that make inappropriate their use in certain applications:

- It compares only time series of the same length.
- It doesn't handle outliers or noise.
- It is very sensitive respect to six signal transformations: shifting, uniform amplitude scaling, uniform time scaling, uniform bi-scaling, time warping and non-uniform amplitude scaling [31].

Dynamic Time Warping (*DTW*) [22] gives more robustness to the similarity computation. By this method, also time series of different length can be compared, because it replaces the one-to-one point comparison, used in Euclidean distance, with a many-to-one (and viceversa) comparison. The main feature of this distance measure is that it allows to recognize similar shapes, even if they present signal transformations, such as shifting and/or scaling (Fig. 4).

Given two time series $T = \{t_1, t_2, \ldots, t_n\}$ and $S = \{s_1, s_2, \ldots, s_m\}$ of length n and m, respectively, an alignment by *DTW* method exploits information contained in a $n \times m$ distance matrix:

$$distMatrix = \begin{pmatrix} d(T_1,S_1) & d(T_1,S_2) & \cdots & d(T_1,S_m) \\ d(T_2,S_1) & d(T_2,S_2) & & \\ \vdots & & \ddots & \\ d(T_n,S_1) & & & d(T_n,S_m) \end{pmatrix} \quad (17)$$

where $distMatrix(i, j)$ corresponds to the distance of ith point of T and jth point of S $d(T_i, S_j)$, with $1 \le i \le n$ and $1 \le j \le m$.

The *DTW* objective is to find the *warping path* $W = \{w_1, w_2, \ldots, w_k, \ldots, w_K\}$ of contiguous elements on *distMatrix* (with $\max(n, m) < K < m + n - 1$, and $w_k = distMatrix(i, j)$), such that it minimizes the following function:

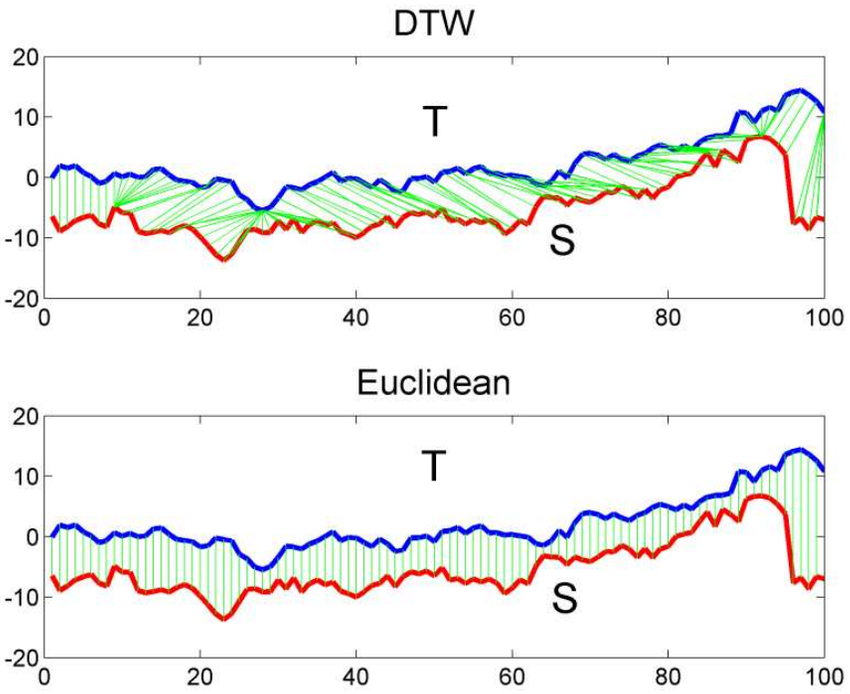

Figure 4. Difference between *DTW* distance and Euclidean distance (green lines represent mapping between points of time series *T* and *S*). The former allows many-to-one point comparisons, while Euclidean point-to-point distance (or one-to-one).

$$DTW(T,S) = \min\left(\sqrt{\sum_{k=1}^{K} w_k}\right) \tag{18}$$

The warping path is subject to several constraints [22]. Given $w_k = (i, j)$ and $w_{k-1} = (i', j')$ with $i, i' \leq n$ and $j, j' \leq m$:

1. **Boundary conditions**. $w_1 = (1,1)$ and $w_K = (n, m)$.
2. **Continuity**. $i - i' \leq 1$ and $j - j' \leq 1$.
3. **Monotonicity**. $i - i' \geq 0$ and $j - j' \geq 0$.

The warping path can be efficiently computed using dynamic programming [9]. By this method, a cumulative distance matrix γ of the same dimension as the *distMatrix*, is created to store in the cell (i, j) the following value (Fig. 5):

$$\gamma(i,j) = d(T_i, S_j) + \min\left\{\gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1)\right\} \tag{19}$$

The overall complexity of the method is relative to the computation of all distances in *distMatrix*, that is $O(nm)$. The last element of the warping path, w_K corresponds to the distance calculated with the *DTW* method.

In many cases, this method can bring to undesired effects. An example is when a large number of point of a time series *T* is mapped to a single point of another time series *S* (Fig. 6a, 7a). A common way to overcome this problem is to restrict the warping path in such a way it has to follow a direction along the diagonal (Fig. 6b, 7b). To do this, we can restrict

the path enforcing the recursion to stop at a certain depth, represented by a threshold δ. Then, the cumulative distance matrix γ will be calculated as follows:

$$\gamma(i,j) = \begin{cases} d(T_i, S_j) + \min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\} & |i-j| < \delta \\ \infty & otherwise \end{cases} \quad (20)$$

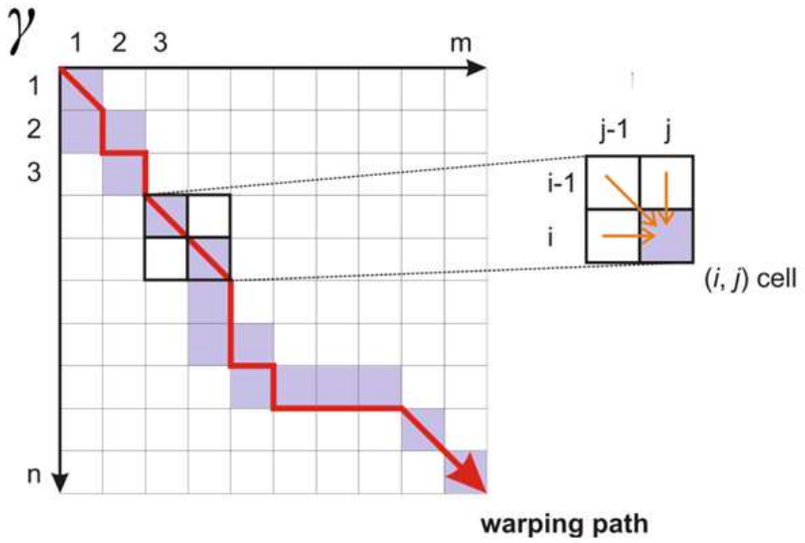

Figure 5. Warping path computation using dynamic programming. The lavender cells corresponds to the warping path. The red arrow indicates its direction. The warping distance at the (*i*, *j*) cell will consider, besides the distance between T_i and S_j, the minimum value among adjacent cells at positions: (*i*-1, *j*-1), (*i*-1, *j*) and (*i*, *j*-1). The Euclidean distance between two time series can be seen as a special case of *DTW*, where path's elements belong to the γ matrix diagonal.

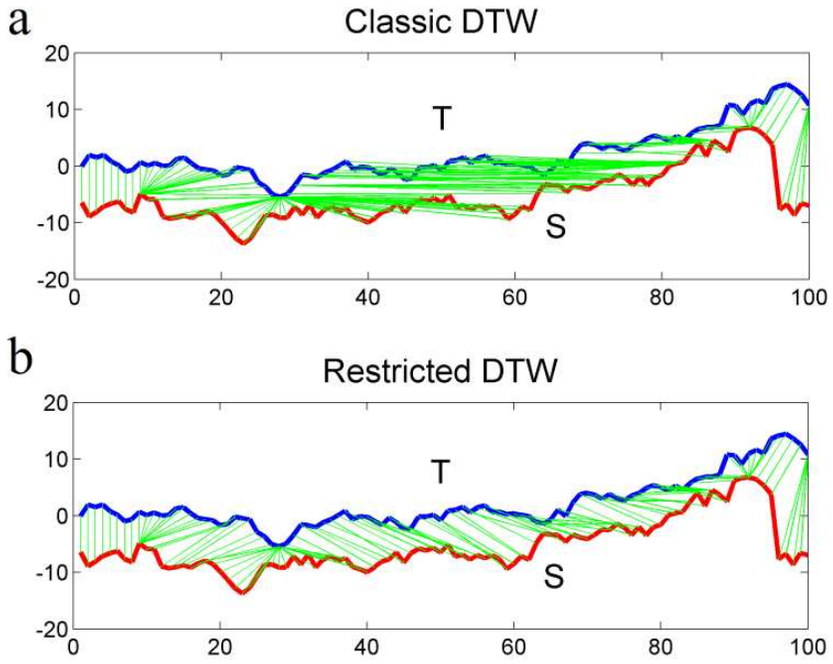

Figure 6. Different mappings obtained with the classic implementation of *DTW* (a), and with the restricted path version using a threshold $\delta = 10$ (b). Green lines represent mapping between points of time series *T* and *S*.

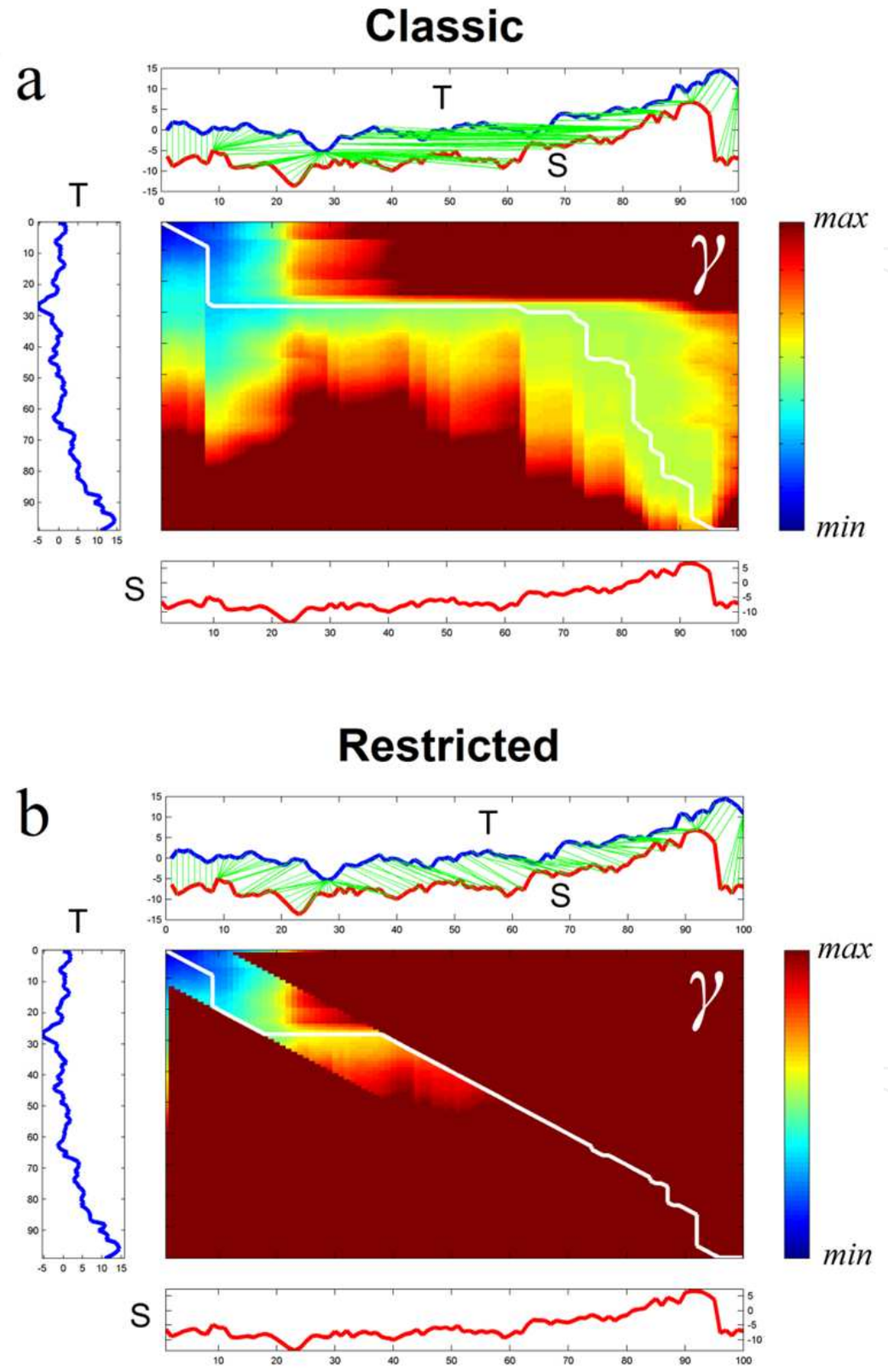

Figure 7. (a) Classic implementation of DTW. (b) Restricted path, using a threshold $\delta = 10$. For each plot (a) and (b): on the center, the warping path calculated on matrix γ. On the top, the alignment of time series T and S, represented by the green lines. On the left, the time series T. On the bottom, the time series S. On the right, the color bar relative to the distance values into matrix γ.

Figure 7a shows the computation of a restricted warping path, using a threshold $\delta = 10$. This constraint, besides limiting extreme or degenerate mappings, allows to speed-up *DTW* distance calculation, because we need to store only distances which are at most δ positions away (in horizontal and vertical direction) from the *distMatrix* diagonal. This reduces the computational complexity to $O((n + m)\delta)$. The above proposed constraint is known also as *Sakoe-Chiba band* (Fig. 8a) [34], and it is classified as global constraint. Another most common global constraint is the *Itakura parallelogram* (Fig. 8b) [18].

Local constraints are subject of research and are different from global constraints [22], because they provide local restrictions on the set of the alternative depth steps of the recurrence function (Eq. 19). For example we can replace Eq. 19 with:

$$\gamma(i,j) = d(T_i, S_j) + \min\left\{\gamma(i-1, j-1), \gamma(i-1, j-2), \gamma(i-2, j-1)\right\} \tag{21}$$

to define a new local constraint.

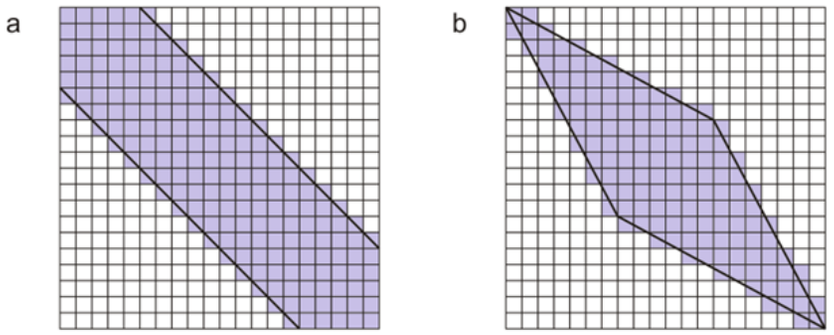

Figure 8. Examples of global constraints: (a) Sakoe-Chiba band; (b) Itakura parallelogram.

3.3. Longest Common SubSequence

Another well known method that takes advantage of dynamic programming to allow comparison of one-to-many points is the *Longest Common SubSequence* (*LCSS*) similarity measure [37]. An interesting feature of this method is that it is more resilient to noise than *DTW*, because allows some elements of time series to be unmatched (Fig. 9). This solution builds a matrix *LCSS* similar to γ, but considering similarity instead of distances. Given the time series T and S of length n and m, respectively, the recurrence function is expressed as follows:

$$LCSS(i,j) = \begin{cases} 0 & i = 0, \\ 0 & j = 0, \\ 1 + LCSS[i-1, j-1] & if \quad T_i = S_j, \\ \max(LCSS[i-1, j], LCSS[i, j-1]) & otherwise \end{cases} \tag{22}$$

with $1 \le i \le n$ and $1 \le j \le m$. Since exact matching between T_i and S_j can be strict for numerical values (Eq. 22 is best indicated for string distance computation, such as the edit distance), a common way to relax this definition is to apply the following recurrence function:

$$LCSS(i,j) = \begin{cases} 0 & i = 0, \\ 0 & j = 0, \\ 1 + LCSS[i-1, j-1] & if \left| T_i - S_j \right| < \varepsilon, \\ \max(LCSS[i-1, j], LCSS[i, j-1]) & otherwise \end{cases} \quad (23)$$

The cell $LCSS(n, m)$ contains the similarity between T and S, because it corresponds to length l of the longest common subsequence of elements between time series T and S. To define a distance measure, we can compute [32]:

$$LCSSdist(T,S) = \frac{n + m + 2l}{m + n} \quad (24)$$

Also for LCSS the time complexity is $O(nm)$, but it can be improved to $O((n + m)\delta)$ if a restriction is used (i.e. when $|i - j| < \delta$).

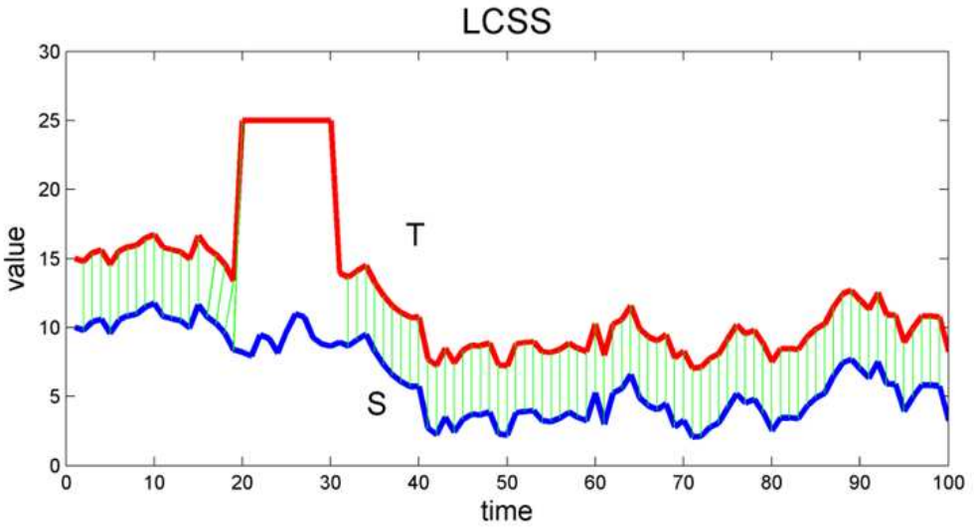

Figure 9. Alignment using *LCSS*. Time series T (red line) is obtained from S (blue line), by adding a fixed value = 5, and further "noise" at positions starting from 20 to 30. In these positions there is no mapping (green lines).

4. Dimensionality reduction techniques

We have already introduced that a key aspect to achieve efficiency, when mining time series data, is to work with a data representation that is lighter than the raw data. This can be done by reducing the dimensionality of data, still maintaining its main properties. An important feature to be considered, when choosing a representation, is the *lower bounding* property.

Given two raw representations of the time series T and S, by this property, after establishing a true distance measure d_{true} for the raw data (such as the Euclidean distance), the distance $d_{feature}$ between two time series, in the reduced space, $R(T)$ and $R(S)$, have to be always less or equal than d_{true}:

$$d_{feature}(R(T), R(S)) \leq d_{true}(T, S) \tag{25}$$

If a dimensionality reduction techniques ensures that the reduced representation of a time series satisfies such a property, we can assume that the similarity matching in the reduced space maintains its meaning. Moreover, we can take advantage of indexing structure such as *GEMINI* (Section 2) [13] to perform speed-up search even avoiding false negative results. In the following subsections, we will review the main dimensionality reduction techniques that preserve the *lower bounding* property.

4.1. DFT

The dimensionality reduction, based on the *Discrete Fourier Transform* (*DFT*) [1], was the first to be proposed for time series. The DFT decomposes a signal into a sum of sine and cosine waves, called *Fourier Series*. Each wave is represented by a complex number known as *Fourier coefficient* (Fig. 10) [29],[32]. The most important feature of this method is the data compression, because the original signal can be reconstructed by means of information carried by the waves with higher Fourier coefficient. The rest can be discarded with no significant loss.

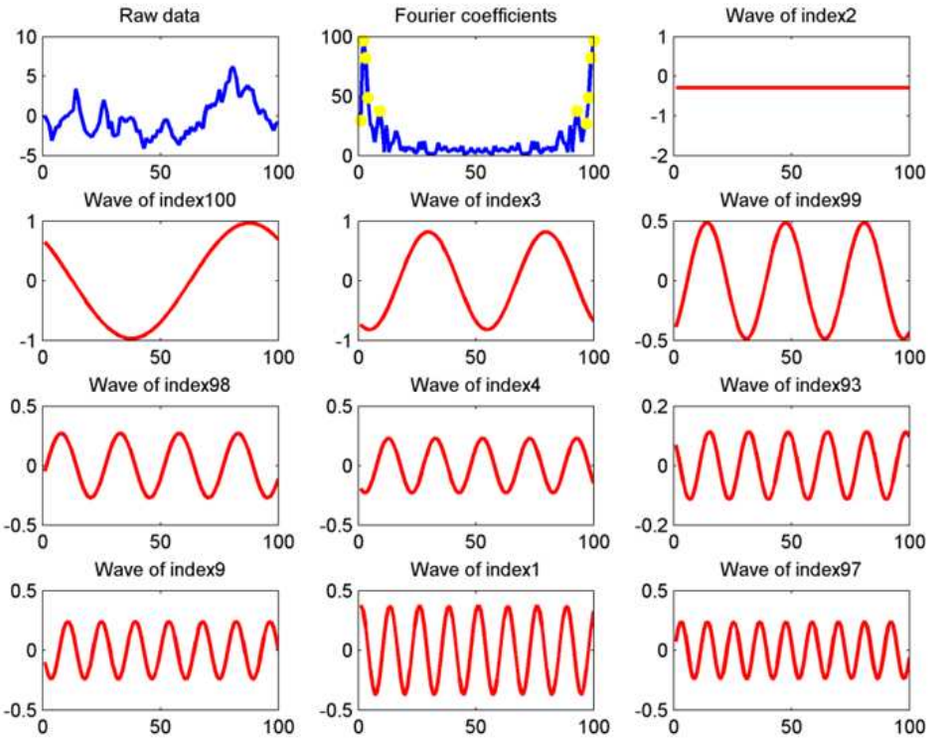

Figure 10. The raw data is in the top-left plot. In the first row, the central plot ("Fourier coefficients" plot) shows the magnitude for each wave (Fourier coefficient). Yellow points are drawn for the top ten highest values. The remaining plots (in order from first row to last, and from left to right) represent the waves corresponding to the top ten highest coefficients in decreasing order, respectively of index {2, 100, 3, 99, 98, 4, 93, 9, 1, 97}, in the "Fourier coefficients" plot.

More formally, given a signal $x = \{x_1, x_2, \ldots, x_n\}$, the n-point *Discrete Fourier Transform* of x is a sequence $X = \{X_1, X_2, \ldots, X_n\}$ of complex numbers. X is the representation of x in the frequency domain. Each wave/frequency X_F is calculated as:

$$X_F = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_i e^{-\frac{2\pi i j F}{n}} \quad (j = \sqrt{-1}) \quad F = 1, \ldots, n \tag{26}$$

The original representation of x, in the time domain, can be recovered by the inverse function:

$$x_i = \frac{1}{\sqrt{n}} \sum_{F=1}^{n} X_F e^{-\frac{2\pi i j F}{n}} \quad (j = \sqrt{-1}) \quad i = 1, \ldots, n \tag{27}$$

The energy $E(x)$ of a signal x is given by:

$$E(x) = \|x\|^2 = \sum_{i=1}^{n} |x_i|^2 \tag{28}$$

A fundamental property of *DFT* is guaranteed by the *Parseval's Theorem*, which asserts that the energy calculated on time series domain for signal x is preserved on frequency domain, and then:

$$E(x) = \sum_{i=1}^{n} |x_i|^2 = \sum_{F=1}^{n} |X_F|^2 = E(X) \tag{29}$$

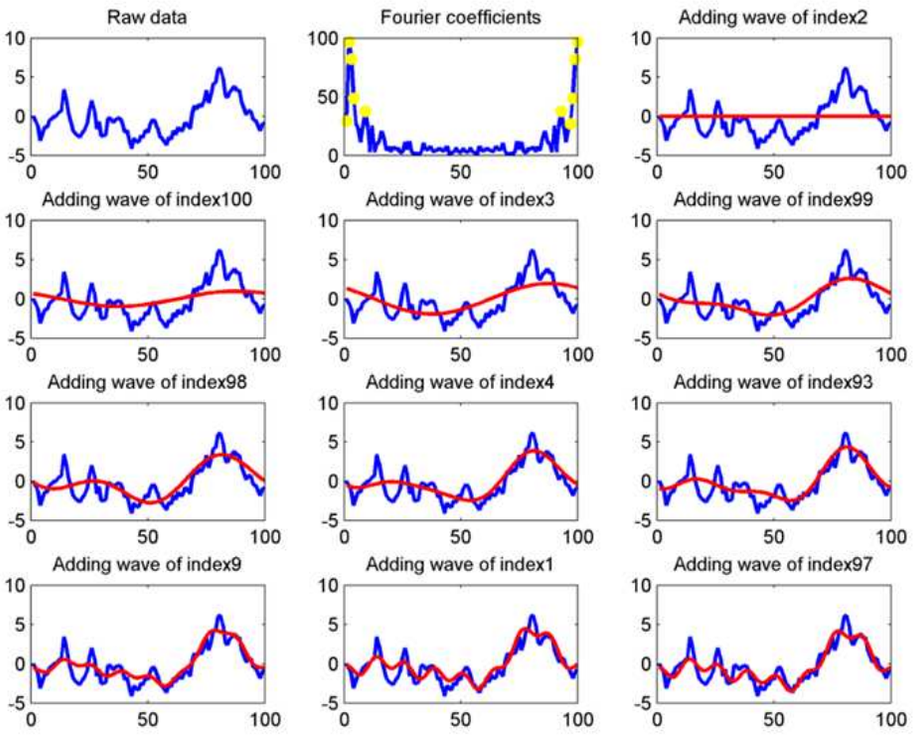

Figure 11. The raw data is in the top-left plot. In the first row, the central plot ("Fourier coefficients" plot) shows the magnitude (Fourier coefficient) for each wave. Yellow points are drawn for the top ten highest values. The remaining plots (in order from first row to last, and from left to right) represent the reconstruction of the raw data using the wave with highest values (of index 2) firstly, then by adding the wave relative to second highest coefficient (of index 100), and so on.

If we use the Euclidean distance (Eq. 12), by this property, the distance $d(x,y)$ between two signals x and y on time domain is the same as calculated in the frequency domain $d(X,Y)$, where X and Y are the respective transforms of x and y. The reduced representation $X' = \{X_1, X_2, \ldots, X_k\}$ is built by only keeping first k coefficients of X to reconstruct the signal x (Fig. 11).

For the Parseval's Theorem we can be sure that the distance calculated on the reduced space is always less than the distance calculated on the original space, because $k \leq n$ and then the distance measured using Eq. 12 will produce:

$$d(X',Y') \leq d(X,Y) = d(x,y) \tag{30}$$

that satisfies the lower bounding property defined in Eq. 25.

The computational complexity of *DFT* is $O(n^2)$, but it can be reduced by means of the *FFT* algorithm [8], which computes the *DFT* in $O(n \log n)$ time. The main drawback of *DFT* reduction technique is the choice of the best number of coefficients to keep for a faithfully reconstruction of the original signal.

4.2. DWT

Another technique for decomposing signals is the *Wavelet Transform* (*WT*). The basic idea of *WT* is data representation in terms of sum and difference of prototype functions, called *wavelets*. The discrete version of *WT* is the *Discrete Wavelet Transform* (*DWT*). Similarly to *DFT*, wavelet coefficients give local contributions to the reconstruction of the signal, while Fourier coefficients always represent global contributions to the signal over all the time [32].

The *Haar* wavelet is the simplest possible wavelet. Its formal definition is shown in [7]. An example of *DWT* based on Haar wavelet is shown in Table 1.

The general *Haar* transform $H_L(T)$ of a time series T of length n can be formalized as follows:

$$A_{L'+1}(i) = \frac{A_{L'}(2i) + A_{L'}(2i+1)}{2} \tag{31}$$

$$D_{L'+1}(i) = \frac{D_{L'}(2i) - D_{L'}(2i+1)}{2} \tag{32}$$

$$H_L(T) = (A_L, D_L, D_{L-1}, \ldots, D_o) \tag{33}$$

where $0 < L' \leq L$, and $1 \leq i \leq n$.

Level (L)	Averages coefficients (A)	Wavelet coefficients (D)
1	10, 4, 6, 6	
2	8, 6	3, 0
3	7	1

Table 1. The *Haar* transform of $T = \{10, 4, 8, 6\}$ depends on the chosen level, and corresponds to merging *Averages coefficients* (column 2) at the chosen level and all *Wavelet coefficients* (column 3) in decreasing order from the chosen level. At level 1 the representation is the same of time series: $H_1(T) = \{10, 4, 6, 6\} + \{\} = \{10, 4, 6, 6\} = T$. At level 2 is $H_2(T) = \{8, 6\} + \{3, 0\} + \{\} = \{8, 6, 3, 0\}$. At level 3 is $H_3(T) = \{7\} + \{1\} + \{3, 0\} = \{7, 1, 3\ 0\}$.

The main drawback of this method is that it is well defined for time series which length n is a power of 2 ($n = 2^m$). The computational complexity of *DWT* using *Haar* Wavelet is $O(n)$. Chan and Fu [7] demonstrated that the Euclidean distance between two time series T and S, $d(T,S)$, can be calculated in terms of their *Haar* transform $d(H(T), H(S))$, by preserving the lower bounding property in Eq. 25, because:

$$d(H(T), H(S)) = \sqrt{2}d(T,S) < d(T,S) \tag{34}$$

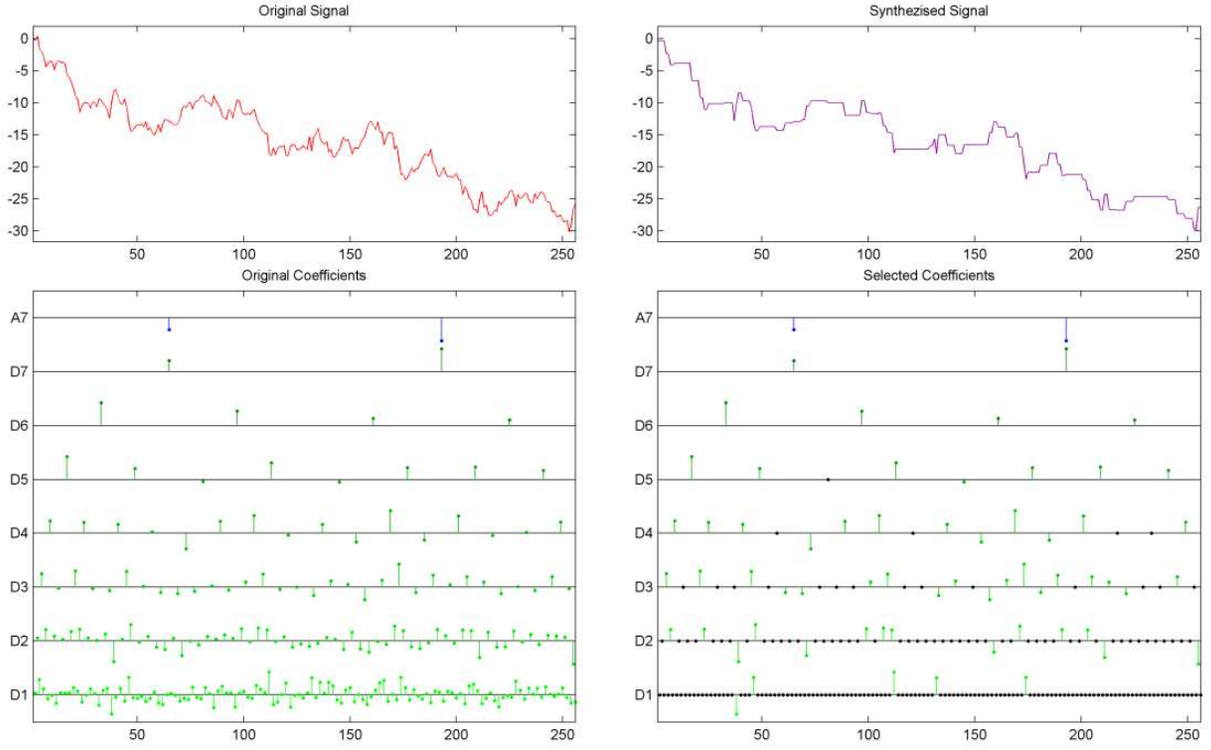

Figure 12. *DWT* using *Haar* Wavelet with *MATLAB Wavelet Toolbox™ GUI tools*. T is a time series of length $n = 256$ and it is shown on the top-left plot (*Original Signal*). On the bottom-left plot (*Original Coefficients*) there are all the A_L, represented by blue stems, and $D_{L'}$ coefficients ($L' < L = 7$), represented by green stems (stems' length is proportional to coefficients value). On the top-right plot, the *Synthesized Signal* by selecting only the 64 biggest coefficients, as reported on the bottom-right plot (*Selected Coefficients*): black points represent unselected coefficients.

4.3. SVD

As we have just seen in Section 2, a *TSDB* with m time series, each of length n, can be represented by a $m \times n$ matrix A (Eq. 1). An important result from linear algebra is that A can always be written in the form [16]:

$$A = UWV^T \tag{35}$$

where U is an $m \times n$ matrix, W and V are $n \times n$ matrices. This is called the *Singular Value Decomposition* (*SVD*) of the matrix A, and the elements of the $n \times n$ diagonal matrix W are the *singular values* w_i:

$$W = \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{pmatrix} \tag{36}$$

V is orthonormal, because $VV^T = V^TV = I_n$, where I_n is the identity matrix of size n. So, we can multiply both sides of Eq. 35 by V to get:

$$AV = UWV^TV \quad \rightarrow \quad AV = UW \tag{37}$$

UW represents a set of n-dimensional vectors $AV = \{X_1, X_2, \ldots, X_m\}$ which are rotated from the original vectors $A = \{x_1, x_2, \ldots, x_m\}$ [29]:

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{pmatrix} = \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_m \end{pmatrix} \begin{pmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{pmatrix} \tag{38}$$

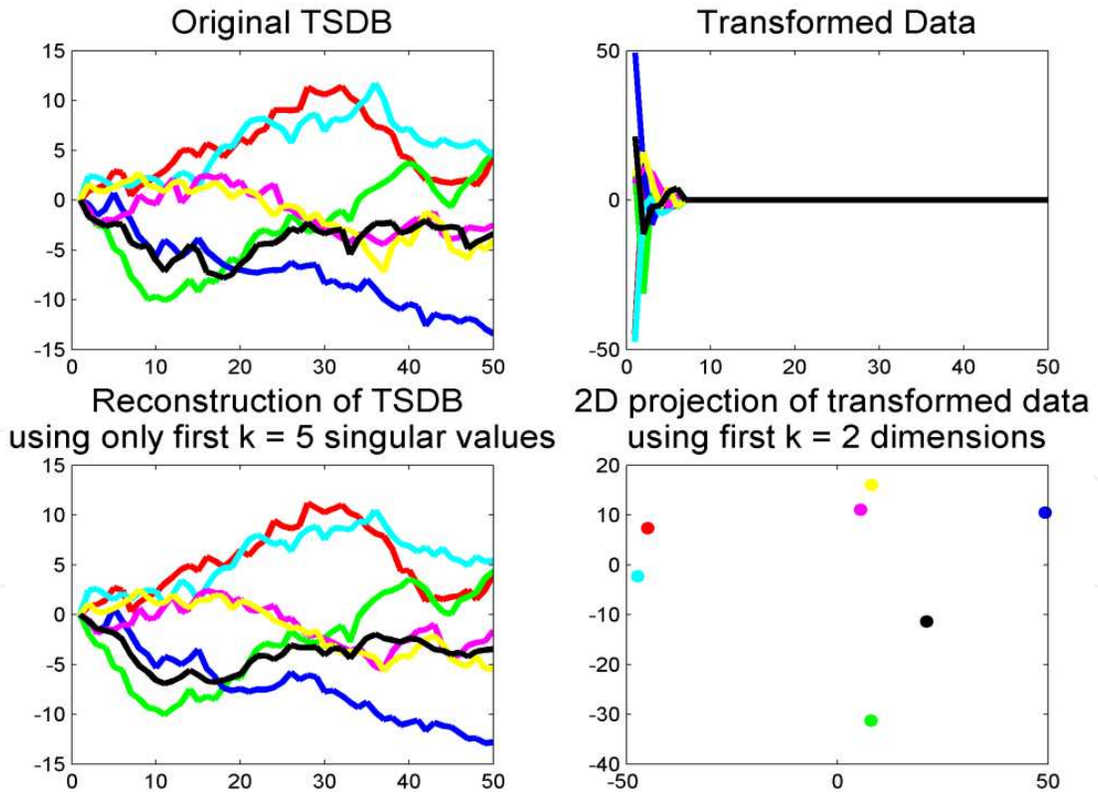

Figure 13. *SVD* for a *TSDB* of m=7 time series of length n=50. It is possible to note in the *transformed data* plot how only first $k < 10$ singular values are significant. In this example we heuristically choose to store only first k=5 diagonal elements of V, and their relative entries in A, U and W, because they represent about 95% of total variance. This permits to reduce space complexity from n to k, still maintaining almost unchanged the information (see the reconstruction on the bottom-left plot).

Similarly to sine/cosine waves for *DFT* (Section 3.1) and to wavelet for *DWT* (Section 3.2), *U* vectors represent basis for *AV*, and their linear combination with *W* (that represents their coefficients) can reconstruct *AV*.

We can perform dimensionality reduction by selecting the first ordered k biggest singular values, and their relative entries in *A*, *V* and *U*, to obtain a new k-dimensional dataset that best fits original data.

SVD is an optimal transform if we aim to reconstruct data, because it minimizes the reconstruction error, but have two important drawbacks: (i) it needs a collection of time series to perform dimensionality reduction (it cannot operate on singular time series), because examines the whole dataset prior to transform. Moreover, the computational complexity is $O(\min(m^2n, mn^2))$. (ii) This transformation is not incremental, because a new data insertion requires a new global computation.

4.4. Dimensionality reduction via PAA

Given a time series T of length n, *PAA* divides it into w equal sized segments t_i ($1 < i \leq w$) and records values corresponding to the mean of each segment $mean(t_i)$ (Fig. 14) into a vector $PAA(T) = \{mean(t_1), mean(t_2), \ldots, mean(t_w)\}$, using the following formula:

$$mean(t_i) = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} t_j \tag{39}$$

When n is a power of 2, each $mean(t_i)$ essentially represents an *Averages coefficient* $A_L(i)$, defined in Section 4.2, and w corresponds in this case to:

$$w = \frac{n}{2^L} \tag{40}$$

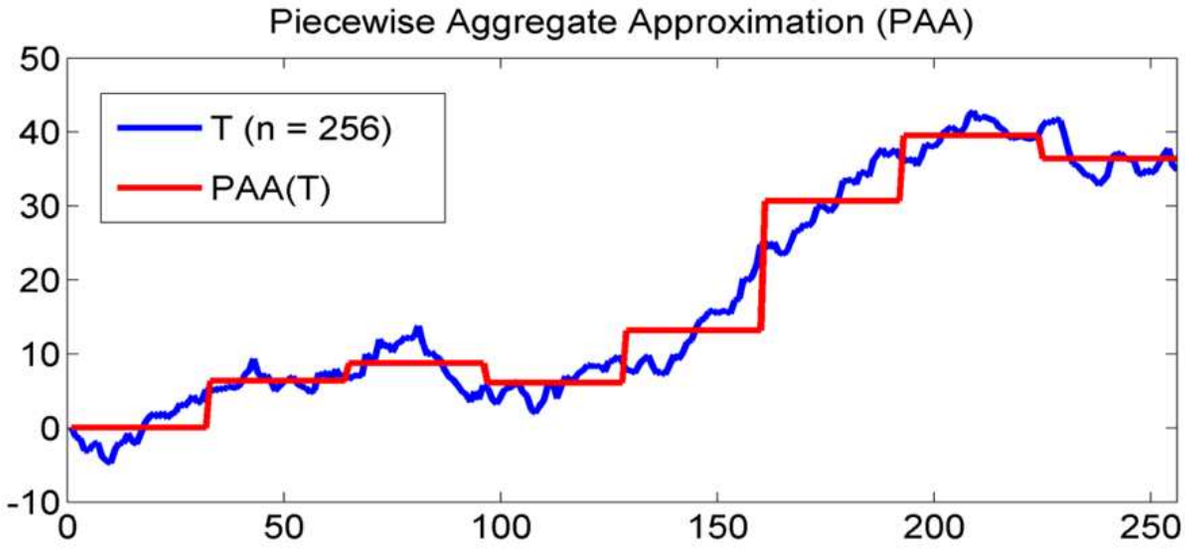

Figure 14. An approximation via *PAA* technique of a time series T of length $n = 256$, with $w = 8$ segments.

The complexity time to calculate the mean values of Eq. 39 is $O(n)$. The *PAA* method is very simple and intuitive, moreover it is strongly competitive with other more sophisticated transforms such as *DFT* and *DWT* [21],[41]. Most of data mining researches makes use of *PAA* reduction for its simplicity. It is simple to demonstrate how the distance on raw representation is bounded below by the distances calculated on *PAA* representation (even using Euclidean distance as reference point), satisfying Eq. 25. A limitation of such a reduction, in some contexts, can be the fixed size of the obtained segments.

4.5. APCA

In Section 4.2 we noticed that not all *Haar* coefficients in *DWT* are important for the time series reconstruction. Same thing for *PAA* in Section 4.4, where not all segment means are equally important for the reconstruction, or better, we sometimes need an approximation with no-fixed size segments. *APCA* is an adaptive model that, differently from *PAA*, allows to define segments of variable size. This can be useful when we find in time series areas of low variance and areas of high variance, for which we want to have, respectively, few segments for the former, and many segments for the latter.

Given a time series $T = \{t_1, t_2, \ldots, t_n\}$ of length n, the *APCA* representation of T is defined as [6]:

$$C = \left\{\left\langle cv_1, cr_1\right\rangle, \ldots, \left\langle cv_M, cr_M\right\rangle\right\}, \quad cr_0 = 0 \tag{41}$$

where cr_i is the last index of the ith segment, and

$$cv_i = mean\left(t_{cr_{i-1}+1}, \ldots, t_{cr_i}\right) \tag{42}$$

To find an optimal representation through the *APCA* technique, dynamic programming can be used [14,30]. This solution requires $O(Mn^2)$ time. A better solution was proposed by Chakrabarti et al. [6], which finds the *APCA* representation in $O(n \log n)$ time, and defines a distance measure for this representation satisfying the lower bounding property defined in Eq. 25. The proposed method bases on *Haar* wavelet transformation. As we have just seen in Section 4.2, the original signal can be reconstructed by only selecting bigger coefficients, and truncating the rest. The segments in the reconstructed signal may have approximate mean values (due to truncation) [6], so these values are replaced by the exact mean values of the original signal. Two aspects to consider before performing APCA:

1. Since *Haar* transformation deals only with time series length $n = 2^p$, we need to add zeros to the end of the time series, until it reaches the desired length.
2. If we held the biggest M Haar coefficients, we are not sure if the reconstruction will return an *APCA* representation of length M. We can know only that the number of segments will vary between M and $3M$ [6]. If the number of segments is more than M, we will iteratively merge more similar adjacent pairs of segments, until we reach M segments.

The algorithm for compute APCA representation can be found in [6].

4.6. Time series segmentation using PLA

As with most computer science problems, representation of data is the key to efficient and effective solutions. A suitable representation of a time series may be *Piecewise Linear Approximation* (*PLA*), where the original points are reduced to a set of segments.

PLA refers to the approximation of a time series T, of length N, using K consecutive segments with K much smaller than n (Fig. 15). This representation makes the storage, transmission and computation of the data more efficient [23]. In the light of it, *PLA* may be used to support clustering, classification, indexing and association rule mining of time series data (e.g. [10]).

The process of time series approximation using *PLA* is known as segmentation and is related to clustering process where each segment can be considered as a cluster [33].

There are several techniques to segment a time series and they can be distinguished into off-line and on-line approaches. In the former approach an error threshold is fixed by the user, while the latter uses a dynamic error threshold that changes, according to specific criteria, during the execution of the algorithm.

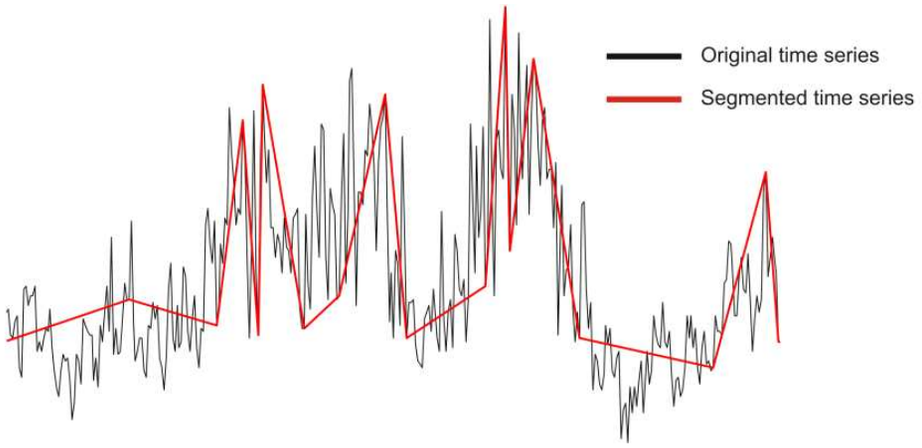

Figure 15. The trend approximation (red line) of the original time series (black line) obtained by *PLA*.

Although off-line algorithms are simple to realize, they are less effective than the on-line ones. The classic approaches to time series segmentation are the sliding window, bottom-up and top-down algorithms. Sliding window is an on-line algorithm and works growing a segment until the error for the potential segment is greater than the user-specified threshold, then the subsequence is transformed into a segment; the process repeats until the entire time series has been approximated by its *PLA* [23]. A way to estimate error is by taking the mean of the sum of the square of vertical differences between the best-fit line and the actual data points. Another commonly used measure of goodness of fit is the distance between the best fit line and the data point furthest away in the vertical direction [23].

In the top-down approach a segment, that represents the entire time-series, is recursively split until the desired number of segment or a error threshold is reached. Dually, the bottom-up algorithm starts from the finest approximation of the time series using $n/2$

segments and merging the two most similar adjacent segments until the desired number of segment or an error threshold is reached.

However, an open question is the choice of best k number of segments. This problem involves a trade-off between compression and accuracy of time series representation. As suggested by Salvador et al. [33], the appropriate number of segments may be estimated using evaluation graph. It is defined as a two dimensional plot where x-axis is the number of segments, while y-axis is a measure of the segmentation error. The best number of segments is provided by the point of maximum curvature, also called "knee", of the evaluation graph (Fig. 16).

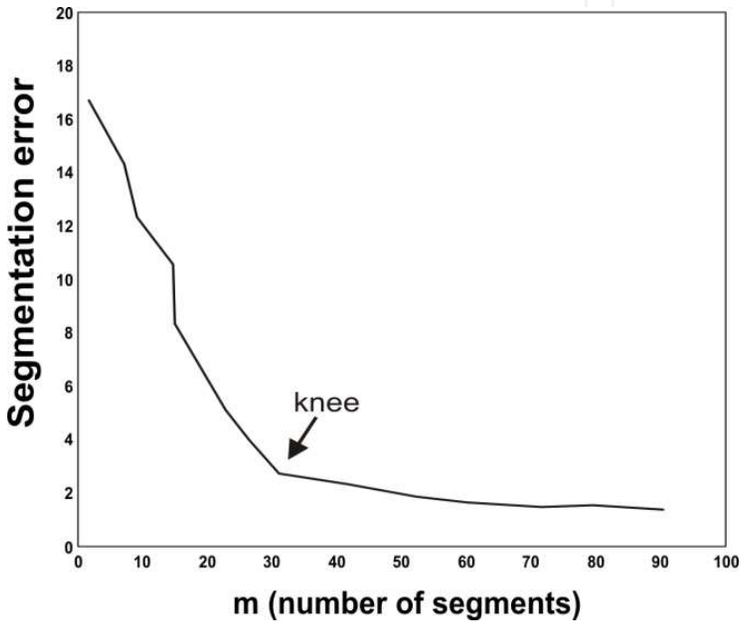

Figure 16. Evaluation graph. The best number of segments is provided by the knee of the curvature.

4.7. Chebyshev Polynomials approximation

By this technique, the reduction problem is resolved by considering the values of the time series T as values of a function f, and approximating it with a polynomial function of degree n which well fits f:

$$P_n(x) = \sum_{i=0}^{n} a_i x^i \tag{43}$$

where each a_i corresponds to coefficients and x^i to the variables of degree i.

There are many possible ways to choose the polynomial: Fourier transforms (Section 4.1), splines, non-linear regressions, etc. Ng and Cai [29] hypothesized that one of the best approaches is the polynomial that minimizes the maximum deviation from the true value, which is called the *minimax* polynomial. It has been shown that the *Chebyshev* approximation is almost identical to the optimal minimax polynomial, and is easy to compute [27]. Thus, Ng and Cai [29] explored how to use the *Chebyshev polynomials* (of the first kind) as a basis

for approximating and indexing n-dimensional ($n \geq 1$) time series. The Chebyshev polynomial $CP_m(x)$ of the first kind is a polynomial of degree m ($m = 0, 1, \ldots$), defined as:

$$CP_m(x) = \cos(m \arccos(x)) \quad x \in [-1, 1] \tag{44}$$

It is possible to compute every $CP_m(x)$ using the following recurrence function [29]:

$$CP_m(x) = 2xCP_{m-1}(x) - CP_{m-2}(x) \tag{45}$$

for all $m \geq 2$ with $CP_0(x) = 1$ and $CP_1(x) = x$. Since Chebyshev polynomials form a family of orthogonal functions, a function $f(x)$ can be approximated by the following Chebyshev series expansion:

$$S_\infty^{CP}(f(x)) = \sum_{i=0}^{\infty} c_i CP_i(x) \tag{46}$$

where c_i refer to the *Chebyshev coefficients*. We refer the reader to the paper [29] for the conversion of a time series, which represents a discrete function, to an interval function required for the computation of Chebyshev coefficients. Given two time series T and S, and their corresponding vectors of Chebyshev coefficients, C_1 and C_2, the key feature of their work is the definition of a distance function d_{Cheb} between the two vectors, that guarantees the lower bounding property defined in Eq. 25. Since it results:

$$d_{cheb}(C_1, C_2) \leq d_{true}(T_1, T_2) \tag{47}$$

the indexing with Chebyshev coefficients admits no false negatives. The computational complexity of Chebishev approximation is $O(n)$, where n is the length of the approximated time series.

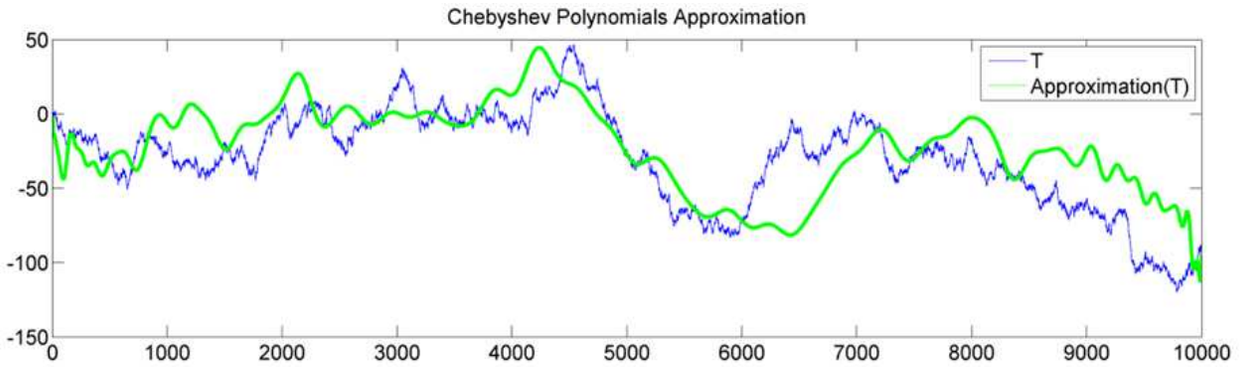

Figure 17. An example of approximation of a time series T of length $n = 10000$ with a Chebyshev series expansion (Eq. 46) where i is from 0 to $k = 100$, using the *chebfun* toolbox for MATLAB (http://www2.maths.ox.ac.uk/chebfun/)

4.8. SAX

Many symbolic representations of time series have been introduced over the past decades. The challenge in this field is to create a real correlation between the distance measure

defined on the symbolic representation, and that defined on original time series. *SAX* is the most known symbolic representation technique on time series data mining, that ensures both a considerable dimensionality reduction, and the lower bounding property, allowing enhancing of time performances on most of data mining algorithm.

Given a time series T of length n, and an alphabet of arbitrary size a, *SAX* returns a string of arbitrary length w (typically $w << n$). The alphabet size a is an integer, where $a > 2$. *SAX* method is *PAA*-based, since it transforms *PAA* means into symbols, according to a defined transformation function.

To give a significance to the symbolic transformation, it is necessary to deal with a system producing symbols with equal probability, or with a Gaussian distribution. This can be achieved by normalizing time series, since normalized time series have generally a Gaussian distribution [26]. This is the first assumption to consider about this technique. However, for data not obeying to this property, the efficiency of the reduction is slightly deteriorated. Given the Gaussian distribution, it is simple to determine the "breakpoints" that will produce a equal-sized areas of probability under the Gaussian curve. What follows gives the principal definitions to understand *SAX* representation.

Definition 1. ***Breakpoints***: A sorted list of numbers $B = \beta_1, \ldots, \beta_{a-1}$ such that the area under a $N(0, 1)$ Gaussian curve from β_i to $\beta_{i+1} = 1/a$ (β_0 and β_a are defined as $-\infty$ and ∞, respectively) (Table 2). For example, if we want to obtain breakpoints for an alphabet of size $a = 4$, we have to compute the first (q_1), the second (q_2), and the third (q_3) quartiles of the inverse cumulative Gaussian distribution, corresponding to the 25%, 50% and 75% of the cumulative frequency: $\beta_1 = q_1$, $\beta_2 = q_2$, $\beta_3 = q_3$.

Definition 2. ***Alphabet***: A collection of symbols $alpha = \alpha_1, \alpha_2, \ldots, \alpha_a$ of size a used to transform mean frames into symbols.

β_i\a	3	4	5	6	7	8
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67
β_3		0.67	0.25	0	-0.18	-0.32
β_4			0.84	0.43	0.18	0
β_5				0.97	0.57	0.32
β_6					1.07	0.67
β_7						1.15

Table 2. A look-up table for breakpoints used for alphabet of size 2 < a < 9.

Definition 3. ***Word***: A PAA approximation $PAA(T) = \{mean(t_1), mean(t_2), \ldots, mean(t_w)\}$ of length w can be represented as a word $SAX(T) = \{sax(t_1), sax(t_2), \ldots, sax(t_w)\}$, with respect to the following mapping function:

$$sax(t_i) = \alpha_j, \quad iff \ \beta_{j-1} \le mean(t_i) < \beta_j \quad (0 < i \le w, \quad 1 < j < a) \tag{39}$$

Lin et al. [26] defined a distance measure for this representation, such that the real distance calculated on original representation is bounded below from it. An extension of *SAX* technique, *iSAX*, was proposed by Shieh and Keogh [35] which allows to get different resolutions for the same word, by using several combination of parameters *a* and *w*.

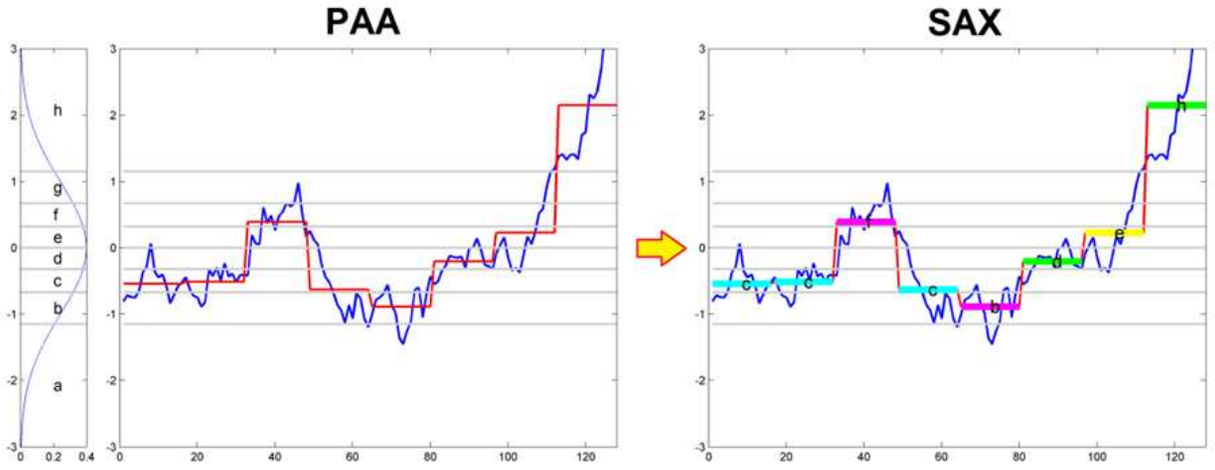

Figure 18. An example of conversion of a time series T (blue line) of length $n = 128$, into a word of length $w = 8$, using an alphabet $alpha = \{a,b,c,d,e,f,g,h\}$ of size $a = 8$. The left plot refers to the Gaussian distribution divided into equal areas of size $1/a$. *PAA* mean frames falling into two consecutive cutlines (gray lines) will be mapped into the corresponding plotted symbol (colored segments). The *PAA* plot shows the *PAA* representation (red line), while *SAX* plot shows the conversion of $PAA(T)$ into the word $SAX(T) = \{c,g,e,g,f,g,a,a\}$. Images generated by MATLAB and code provided by *SAX* authors [26].

Author details

Placido Montalto, Marco Aliotta and Andrea Cannata
Istituto Nazionale di Geofisica e Vulcanologia, Osservatorio Etneo, Sezione di Catania, Catania, Italy

Carmelo Cassisi* and Alfredo Pulvirenti
Università degli studi di Catania, Dipartimento di Matematica e Informatica, Catania, Italy

5. References

[1] Agrawal, Faloutsos, Swami (1993). "Efficient similarity search in sequence databases". *Proc. of the 4th Conference on Foundations of Data Organization and Algorithms*, pp. 69–84.

[2] Bailey, Elkan (1995). "Unsupervised learning of multiple motifs in biopolymers using expectation maximization". *Machine Learning Journal 21*, 51–80.

[3] Beckmann, Kriegel, Schneider, Seeger (1990). "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles". *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Atlantic City, NJ, pp. 322-331.

[4] Bentley (1975). "Multidimensional binary search trees used for associative searching". *Communications of ACM*. Vol. 18, pp. 509-517.

* Corresponding Author

[5] Cantone, Ferro, Pulvirenti, Reforgiato (2005). "Antipole tree indexing to support range search and k-nearest neighbour search in metric spaces". *IEEE Transactions on Knowledge and Data Engineering*. Vol. 17, pp. 535-550.

[6] Chakrabarti, Keogh, Mehrotra, Pazzani (2002). "Locally adaptive dimensionality reduction for indexing large time series databases". *ACM Trans. Database Syst.* 27, 2, pp 188-228.

[7] Chan, Fu (1999). "Efficient time series matching by wavelets". In *proceedings of the 15th IEEE Int'l Conference on Data Engineering*. Sydney, Australia, Mar 23-26. pp 126-133.

[8] Cooley, Tukey (1965). "An algorithm for the machine calculation of complex Fourier series", *Math. Comput.* 19, 297–301.

[9] Cormen, Leiserson, Rivest (1990). *Introduction to Algorithms*. The MIT Press, McGraw-Hill Book Company.

[10] Di Salvo, Montalto, Nunnari, Neri, Puglisi (2012). "Multivariate time series clustering on geophysical data recorded at Mt. Etna from 1996 to 2003". *Journal of Volcanology and Geothermal Research*.

[11] Ding, Trajcevski, Scheuermann, Wang, Keogh (2008). "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures". *VLDB*.

[12] Duda, Hart, Stork (2001). *Pattern Classification*. John Wiley & Sons.

[13] Faloutsos, Ranganthan, Manolopoulos (1994). "Fast subsequence Matching in Time-Series Databases". *SIGMOD Conference*.

[14] Faloutsos, Jagadish, Mendelzon, Milo (1997). "A signature technique for similarity-based queries". In *Proceedings of the SEQUENCES 97* (Positano-Salerno, Italy).

[15] Fink, Pratt (2004). "Indexing of compressed time series**".** In Mark Last, Abraham Kandel, and Horst Bunke, editors, *Data Mining in Time Series Databases*, pages 43-65. World Scientific, Singapore.

[16] Golub, Van Loan (1996). *Matrix Computations*, 3rd edition. Baltimore, MD: Hopkins University Press.

[17] Han and Kamber (2005). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, CA.

[18] Itakura (1975). Minimum prediction residual principle applied to speech recognition. *IEEE Trans Acoustics Speech Signal Process*. ASSP 23:52–72

[19] Keogh, Chakrabarti, Pazzani, Mehrotra (2000). "Dimensionality reduction for fast similarity search in large time series databases". *Journal of Knowledge and Information Systems*.

[20] Keogh, Chu, Hart, Pazzani (2001a). An Online Algorithm for Segmenting Time Series. In Proc. IEEE Intl. Conf. on Data Mining, pp. 289-296, 2001.

[21] Keogh, Kasetty (2002). "On the need for time series data mining benchmarks: a survey and empirical demonstration". *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 102-111.

[22] Keogh, Ratanamahatana (2002). "Exact indexing of dynamic time warping". In *proceedings of the 26th Int'l Conference on Very Large Data Bases*. Hong Kong. pp 406-417.

[23] Keogh, Chu, Hart, Pazzani (2004). "Segmenting time series: a survey and novel approach". In: Last, M., Kandel, A., Bunke, H. (Eds.), *Data mining in time series database*. World Scientific Publishing Company, pp. 1-21.

[24] Korn, Jagadish, Faloutsos (1997). "Efficiently supporting ad hoc queries in large datasets of time sequences". *Proceedings of SIGMOD '97*, Tucson, AZ, pp 289-300.

[25] Lawrence, Altschul, Boguski, Liu, Neuwald, Wootton (1993). "Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment". *Science 262*, 208–14.

[26] Lin, Keogh, Wei, Lonardi (2007). "Experiencing SAX: a novel symbolic representation of time series". Data Mining Knowledge Discovery, 15(2).

[27] Mason, Handscomb (2003). *Chebyshev Polynomials*. Chapman & Hall.

[28] Mueen, Keogh, Zhu, Cash, Westover (2009). "Exact Discovery of Time Series Motifs". *SDM 2009*.

[29] Ng, Cai (2004): "Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials". *SIGMOD 2004*.

[30] Pavlidis (1976). "Waveform segmentation through functional approximation". *IEEE Trans.Comput. C-22*, 7 (July).

[31] Perng, Wang, Zhang, Parker (2000). "Landmarks: a newmodel for similarity-based pattern querying in time series databases". *Proc.2000 ICDE*, pp. 33–42.

[32] Ratanamahatana, Lin, Gunopulos, Keogh, Vlachos, Das (2010): "Mining Time Series Data". *Data Mining and Knowledge Discovery Handbook*, pp. 1049-1077.

[33] Salvador, Chan, (2004). "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms". *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, 2004, pp. 576-584.

[34] Sakoe, Chiba (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans Acoustics Speech Signal Process*. ASSP 26:43–49

[35] Shieh and Keogh (2008). "iSAX: Indexing and Mining Terabyte Sized Time Series". *SIGKDD*, pp 623-631.

[36] Tompa, Buhler (2001). "Finding motifs using random projections". *In proceedings of the 5th Int'l Conference on Computational Molecular Biology*. Montreal, Canada, Apr 22-25. pp 67-74.

[37] Vlachos, Kollios, Gunopulos (2002). "Discovering similar multidimensional trajectories". *Proc. 2002 ICDE*, pp. 673–684.

[38] Von Storch, Zwiers (2001). *Statistical analysis in climate research.* Cambridge Univ Pr. ISBN 0521012309.

[39] Wang, Ye, Keogh, Shelton (2008). "Annotating Historical Archives of Images". *JCDL 2008*.

[40] Yang, Shahabi (2004). "A PCA-based similarity measure for multivariate time series". In Proceedings of the 2nd ACM international workshop on Multimedia database, pp. 65-74.

[41] Yi and C. Faloutsos (2000). "Fast Time Sequence Indexing for Arbitrary Lp Norms". *VLDB*.