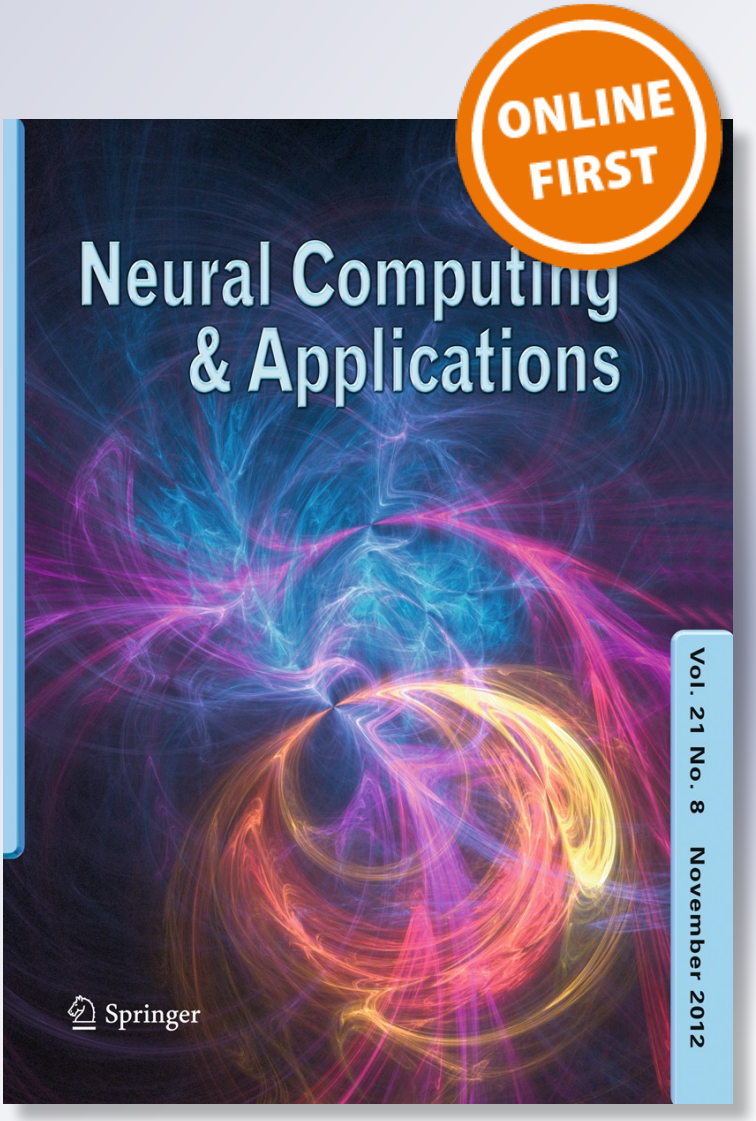
Piecewise evolutionary segmentation for feature extraction in time series models

Thomas J. Glezakos, Theodore A. Tsiligiridis & Constantine P. Yialouris

Neural Computing and Applications

ISSN 0941-0643

Neural Comput & Applic
DOI 10.1007/s00521-012-1212-y



Springer

Neural Comput & Applic
DOI 10.1007/s00521-012-1212-y

ORIGINAL ARTICLE

Piecewise evolutionary segmentation for feature extraction in time series models

Thomas J. Glezakos · Theodore A. Tsiligiridis · Constantine P. Yialouris

Received: 23 September 2011 / Accepted: 25 September 2012

Abstract The design, development and implementation of an innovative system utilized in feature extraction from time series data models is described in this manuscript. Achieving to design piecewise segmentation patterns on the time series in an evolutionary fashion and use them in order to produce fitter secondary data sets, the developed system adapts itself to the nature of the problem each time and finally elects an optimally parameterized classifier (artificial neural network or support vector machine), along with the fittest time series segmentation pattern. The application of the system onto two different problems involving time series data analysis and requiring predictive and classification capabilities (torrential risk assessment and plant virus identification, respectively), reveals that the proposed methodology was crucial in finding the optimum solution for both problems. Piecewise evolutionary segmentation time series model analysis, utilized by the accompanying software tool, succeeded in controlling the dimensionality and noise inherent in the initial raw time series information. The process eventually proposes a segmentation pattern for each problem, enhancing the potential of the corresponding classifier.

T. J. Glezakos (✉) · T. A. Tsiligiridis · C. P. Yialouris
Informatics Laboratory, Division of Informatics, Mathematics and Statistics, Department of Science, Agricultural University of Athens, Iera Odos 75, 11855 Athens, Hellas
e-mail: t_glezakos@yahoo.com

T. A. Tsiligiridis
e-mail: tsili@aua.gr

C. P. Yialouris
e-mail: yialouris@aua.gr

Keywords Evolutionary computing · Machine learning · Artificial neural networks · Support vector machines · Plant virus identification · Torrential risk management

1 Introduction

The problem we focus on in this paper is to propose a competent method by which a time series sequence of real numbers may be effectively represented in lower dimensionality space. Stemming from a piecewise segmentation of the time series, the method we have designed keeps crucial information intact, while transforming the rest such that the identifying mechanism exhibits good results, after testing a number of candidate representations and choosing the fittest. An evolutionary procedure under which the segmentation scheme has been wrapped up fills the data pool with possibly fit representations of the time series and the fittest is finally elected. Historical and temporal time series databases may well benefit from such a facility, whether the time series analysis corresponds to classification or regression problems. Specifically,

- environmental time series databases, including topographical, weather and anaglyph data for the prediction of certain phenomena [1] or
- biological time series databases, including sensor data in need of classification matching, especially in the cases where the time series is really excessively wide [2]

will benefit from the representation of the initial information in lower dimensionality and from the generation of a series of more potent evolutionary data, specifically fit for the chosen classification or regression tool each time. Effective feature extraction of time series in such databases

Published online: 17 October 2012

is essential for various undertaken tasks implying data mining in identification and forecasting problems.

1.1 Computational intelligence in time series analysis

During the recent years, a major boost in evolutionary computing and computational intelligence is occurring, including various and diverse tools. Various proposals in the literature state that genetic algorithm (GA), artificial neural network (ANN) and support vector machine (SVM) models have once again attracted the attention of analysts, experts and consultants, mainly due to the fact that the hardware and know-how we now have at our disposal is capable of simulating such procedures.

The ANN concept was developed in an attempt to simulate the function of the human brain which, as a complicated biological machine, works in parallel and is able to solve innumerable kinds of problems. Basically, an ANN is a software device consisting of simple processing units, the neurons, embedded in layers, interconnected and working in parallel. Each neuron is only aware of the signals it receives from and sends to other connected neurons. The training of the network and the flow of information from layer to layer enables it to learn by example through iterations. Thus, learning is the process of adapting the neurons' connection weights in response to stimuli presented as inputs requiring the presence of a known output. This process aims to adjust the strengths of the connections between the processing elements in a search for the best synaptic weight vector. Effective training enhances the system's generalization, enabling it to be applied successfully to problems incorporating large numbers of variables. The SVM, on the other hand, stems from the maximum margin classifiers theory which has become the base for various highly regarded state-of-the-art classification training algorithms. The margin is the distance from a hyper-plane separating the classes to the nearest point in the data set. The maximum margin requirement has the advantage that it produces unique solutions for linearly separable problems, while offering robustness against noise in data. SVMs utilize kernel functions based on the structured risk minimization (SRM) principle and their effectiveness emanates from the minimization of the upper bound of the generalization error. Details for the aforementioned models already known to the AI research community may be found in [3] and [4].

Several instances of such tools are included into a vast list of novel research methodologies [5–8] starting to play a major role in time series representation. ANNs are reported as cost effective methods for achieving good results with time series while utilizing meta–heuristic methods [9–14]. In other cases, novel hybrid models are engaged, which include combinations of auto regressive integrated moving average (ARIMA) models and ANNs [15], rough set back propagation [16], recursive structured set of multi-layered perceptrons (MLPs) or auto-deterministic networks [17]. Alternatively, the research has resorted to SVMs which have been reported as offering a good alternative to neural networks, improving the generalization performance while achieving better overall results, especially when combined with genetically driven pre-processing [18, 19]. Some researchers utilized the modified SVM models such as least squares SVM normal [20] and recurrent [21], minimum class variance SVMs [22] or radial-based function (RBF) neural networks [23, 24], though the mainstream focuses on normal SVMs as the most prominent constructors, both as regards to regression and classification problems. Also, GAs and variations of evolutionary programming have been used effectively [25–28] in order to utilize time series segmentation and facilitate effective representation.

1.2 Time series segmentation

The significance of time series analysis has already been stressed out as crucial in many disciplines as diverse as energy, finance, econometrics, biology, clinical medicine, meteorology, hydrology and hydraulics, forestry, plant and animal production and agriculture. Environmental modelling has also in many cases engaged time series historical information so as to facilitate decision-making. A time series is generated by the mechanism of a phenomenon producing sequences of vectors, measured at successive constant time intervals, each of which corresponds to either a given class or a value. The analysis encompasses processes to unveil the underlying context of the mechanism producing the initial data, either in the scope of identifying the target class or predicting future states of the system, always based on the knowledge of already measured and quantified past events. This kind of procedure underlines the fact that past orderly measurements give enough input to reproduce the phenomenon in question, thus it aims at modelling the mechanism responsible for the production of the output [29, 30] over time.

A crucial step in the context of feature extraction for such models is time series representation [31]. Various such algorithms have been proposed, including, but not limited to, sampling, averaging and exponential smoothing [32–34], symbolic mappings [35, 36] and Fourier transformations [37, 38], while various autoregressive conditional heteroscedasticity (ARCH) models, normal or generalized, have already appeared as an alternative to traditional econometric analyses [39–41].

Perhaps one of the most ubiquitous representation schemes in time series analysis is provided by piecewise

linear representation (PLR) models, according to which a time series of length n is approximated by K straight lines, where K is typically much smaller than n. According to Keogh et al. [42], the algorithms which input time series data and return their piecewise linear representations are categorized as *time series segmentation algorithms*. In order for an algorithm to fall under this category, three assumptions must be met while producing the best representation: it must use an arbitrary number of segments, while the maximum and average error of all segments is less than an arbitrary threshold. During the last decade, there has been a significant increase in the use of segmentation algorithms for time series analysis. Furthermore, due to the method's wide acceptance, a lot of variations have arisen aiming primarily at the enhancement of the resulting representation. Ding et al. [43] use segment radian errors in order to enhance data compression and time series representation. A segment radian is defined as the radian of the included angle created by the segment and the time axis, while radian error is the value of radian difference between two adjacent segments. The modified algorithm identifies crucial time series data points according to a pre-defined cumulative radian error threshold. Guerrero et al. [44] approach the segmentation process as a multi-objective optimization problem (MOOP). Along with the final approximation error obtained, they introduce the number of segments in the final representation as another quality metric over the final results. The objective functions that formalize the MOOP are in conflict since the error value decreases as the number of segments rises due to finer approximations. The researchers use an evolutionary algorithm to solve the corresponding MOOP. Tseng et al. [45] proposed a segmentation approach which combined the clustering technique, discrete wavelet transformation and genetic algorithms in a process to find the segmentation points for deriving appropriate patterns. Wang and Wang [46] utilized a structure-adaptive k piecewise segmentation procedure, according to which regression lines deriving from the local minimum and maximum of the series represented the segmentations.

1.3 Piecewise evolutionary segmentation

The recent increase in the use of evolutionary programming and computational intelligence tools, along with the need for better time series representation schemes, has motivated us to combine their power with PLR so as to design an innovative technique for time series segmentation, used as data pre-processing procedure. The piecewise evolutionary segmentation (PES) method for time series representation that we propose in the present work is characterized by the utilization of a vibrant segmental methodology, adapting the width, the number and the contents of the segments to the data explored.

PES may be registered as a PLR variant. Just like the latter, it is an algorithm which performs representation by segmentation, aiming to extract the most crucial time series data. Another similarity is that it utilizes an approximation procedure for the segments designed. On the other hand, it differs from PLR as regards to the approximation procedure, which in the case of PES depends on a genetic algorithm. While PLR utilizes a pre-defined number of segments, PES adapts its segments to the data explored, behaviour which is dictated by the production of a number of generations consisting of segmentation schemes. Each scheme is genetically produced and approximated, while the algorithm evaluates each segmentation performance embedding ANN and SVM classifiers. In this context, PES differs from our previous research work [47], in that it has matured into an integrated software tool which elects the best fit evolutionary data set, as well as the most prominent classifier, that is, ANN or SVM, for each given case. It also swerves from other PLR variations [43, 44], in that it produces and evaluates time series segmentation schemes in an evolutionary fashion, that is, the representation is based on an innovative evolutionary segmentation of the time series.

Thus, an evolutionary segmentation model pre-processes the initial time series information and produces fitter secondary data sets used in the supervised training and testing of ANN and SVM classifiers. The design of the evolutionary segmentation patterns on the input plane is materialized via a specially constructed GA. A population of *chromosomes* is produced in each generation (which we will call *trainers* from now on) that maps part of their scheme onto the initial time series information. This depiction is achieved according to a mechanism dictated by certain genes inside the trainer. Thus, a multitude of possible secondary data sets are produced and subsequently used in turn for the supervised training of a computational intelligence classifier. Its performance, measured at the testing phase, is quantified as a fitness score assigned to the corresponding trainer. Successive generations continue until the fittest trainer (i.e. the best segmentation scheme, the one that is responsible for the best performance) and the best classifier are elected.

The remaining part of the paper is organized as follows: in Sect. 2, the PES model is presented, analyzed and explained through some working examples. Section 3 describes the process workflow, along with the evolutionary

internal functions of mating, selection, crossover and mutation. The applications on which the proposed methodology was applied and the yielding results are presented in Sect. 4, while conclusions are provided in Sect. 5.

2 Piecewise evolutionary segmentation model

Assume that for a given problem:

- The input array $\mathbf{X} = [x_{mn}],\ x_{mn} \in \mathbf{R},\quad m = 1, 2, \ldots, M,\ n = 1, 2, \ldots, N$ is described by M records of time series data, each of which comprises of N time stamps, namely:

$$\mathbf{X} = [\mathbf{X}_m]_{m=1(1)M} = [x_{m1}, x_{m2}, \ldots, x_{mN}]_{m=1(1)M} \tag{1}$$

- The corresponding output matrix $\mathbf{Y} = [y_{mp}],\ y_{mp} \in \{0, 1\},\ m = 1, 2, \ldots, M,\ p = 1, 2, \ldots, P$ is described by M records, each of which comprises of P patterns, namely:

$$\mathbf{Y} = [\mathbf{Y}_m]_{m=1(1)M} = [y_{m1}, y_{m2}, \ldots, y_{mP}]_{m=1(1)M} \tag{2}$$

- The matrix of trainers $\boldsymbol{\Omega} = [\omega_{qn}],\ \omega_{qn} \in \{0, 1\},\ q = 1, 2, \ldots Q,\ n = 1, 2, \ldots N + 2$ is described by Q records (trainers), each of which comprises of $N + 2$ elements, namely:

$$\boldsymbol{\Omega} = \left[\boldsymbol{\Omega}_q\right]_{q=1(1)Q} = \left[\omega_{q1}, \omega_{q2}, \ldots, \omega_{qN} \vdots \omega_{qN+1}, \omega_{qN+2}\right]_{q=1(1)Q} \tag{3}$$

Definition 1 A trainer consists of randomly chosen binary genes ordered into two blocks, the *activation block* and the *core block*, having the form: $\boldsymbol{\Omega}_q = (\omega_{q1}, \omega_{q2}, \ldots, \omega_{qN} \vdots \omega_{qN+1}, \omega_{qN+2});\quad q = 1, 2, \ldots, Q.$

Table 1 presents the structure of a trainer through a working example. As it will be seen, the core of the trainer stands as a behavioural mechanism defining a set of rules so as to dictate actions taken by the activation block, the rest of the trainer, towards the raw data that it manipulates each time.

Table 1 Example of the structure of a trainer (a) divided in the activation and core blocks

Activation block										Core block		
1	0	0	1	1	0	0	1	0	1	1	0	(a)
Ω^3			Ω^1	Ω^3			Ω^2		Ω^1			(b)
44	32	17	8	8	12	1	18	30	48			(c)
X^3			X^1	X^3			X^2		X^1			(d)

The activation block is divided by the evolutionary algorithm into five trainer segments (b), with which it manipulates a 10-element initial raw time series record (c), designing a segmentation scheme consisting of five consecutive time series segments (d)

Definition 2 The N binary genes of the activation block of a trainer $\boldsymbol{\Omega}_q; q = 1, 2, \ldots Q$ define a segment $\boldsymbol{\Omega}_q^k; k \in \mathbf{N}$ as follows:

$$\boldsymbol{\Omega}_q^1 = \underbrace{1}_{1-\text{element}},\ \boldsymbol{\Omega}_q^2 = \underbrace{1\ \ 0}_{2-\text{elements}},\ \boldsymbol{\Omega}_q^3 = \underbrace{1\ \ 0\ \ 0}_{3-\text{elements}}, \ldots\ldots,$$
$$\boldsymbol{\Omega}_q^k = \underbrace{1\ \ 0\ \ 0\ \ \cdots\ \ 0}_{k-\text{elements}} \tag{4}$$

namely, as a k-set of binary elements, the first of which is 1 and all the rest are 0.

It is important to note that since the first digit represents the initialization of the phenomenon, the first gene of the activation block for each trainer should always be 1.

In the same way, each of the M records $\mathbf{X}_m = [x_{m1}, x_{m2}, \ldots, x_{mN}]$ of the matrix $\mathbf{X} = [x_{mn}]\ m = 1, 2, \ldots, M,\ n = 1, 2, \ldots, N$ is partitioned into equal number of time series segments, with each one $\mathbf{X}_m^k; k \in \mathbf{N}$ defined by the corresponding trainer segment $\boldsymbol{\Omega}_q^k$. For example, omitting for simplicity the index m, the activation block of the trainer shown in Table 1 consists of the following five consecutive trainer segments: $\boldsymbol{\Omega}^3\ \boldsymbol{\Omega}^1\ \boldsymbol{\Omega}^3\ \boldsymbol{\Omega}^2\ \boldsymbol{\Omega}^1$. Therefore, the corresponding five consecutive segments for the initial row data will be given by: $\mathbf{X}^3\ \mathbf{X}^1\ \mathbf{X}^3\ \mathbf{X}^2\ \mathbf{X}^1$ or equivalently by: [44 32 17] [8] [8 12 1] [18 30] [48].

To complete our notation, it is necessary to include the number of segments, say r, of the above record, say m, as well as the number of elements in each segment, say $k_j; j = 1(1)r$. Thus, denoting the segments identified previously with $\boldsymbol{\Omega}_m^{k_1}\ \boldsymbol{\Omega}_m^{k_2}\ \boldsymbol{\Omega}_m^{k_3}\ \boldsymbol{\Omega}_m^{k_4}\ \boldsymbol{\Omega}_m^{k_5}$ and $\mathbf{X}_m^{k_1}\ \mathbf{X}_m^{k_2}\ \mathbf{X}_m^{k_3}\ \mathbf{X}_m^{k_4}\ \mathbf{X}_m^{k_5}$, respectively, we may easily verify that the number of segments defined in the record m is $r = 5$, with $k_j; j = 1(1)5$, and $k_1 = 3, k_2 = 1, k_3 = 3, k_4 = 2$, and $k_5 = 1$.

A trainer comprised of two blocks, the activation block, segmented as defined above, as well as the core block, is presented as follows:

$$\boldsymbol{\Omega} = \left[\boldsymbol{\Omega}_q\right]_{q=1(1)Q} = \left[\omega_{q1}, \omega_{q2}, \ldots, \omega_{qN} \vdots \omega_{qN+1}, \omega_{qN+2}\right]_{q=1(1)Q}$$
$$= \left[\boldsymbol{\Omega}_q^{k_1}\ \boldsymbol{\Omega}_q^{k_2}, \ldots, \boldsymbol{\Omega}_q^{k_r} \vdots \omega_{qN+1}, \omega_{qN+2}\right]_{q=1(1)Q} \tag{5}$$

with $\sum_{j=1}^{r} k_j = N$ and all $k_j, N, r, q \in \mathbf{N}$. Thus, each trainer $\boldsymbol{\Omega}_q = (\omega_{q1}, \omega_{q2}, \ldots, \omega_{qN} \vdots \omega_{qN+1}, \omega_{qN+2})$, is combined with the

input matrix X, through the operator '$\circ$' in order to produce the matrix $\Xi^q = \left[\xi^q_{mk}\right]$ $\forall m = 1,2,\ldots,M$, $k = 1,2,\ldots k_r$, $q = 1,2,\ldots,Q$, with $M, k_r, Q \in N$, as follows:

$$\begin{aligned}\Xi^q \leftarrow X \circ \Omega_q : \left[\xi^q_{mk}\right] &= [x_{mk}] \circ \Omega_q \\ &= (x_{m1}, x_{m2}, \ldots, x_{mN})_{m=1(1)M} \circ (\omega_{q1}, \omega_{q2}, \ldots, \omega_{qN} \vdots \omega_{qN+1}, \omega_{qN+2}) \\ &= \left[X_m^{k_1} X_m^{k_2} \cdots X_m^{k_r}\right]_{m=1(1)M} \circ \left[\Omega_q^{k_1}\ \Omega_q^{k_2}\ \cdots \Omega_q^{k_r} \vdots \omega_{qN+1}, \omega_{qN+2}\right] \\ &= \left[X_m^{k_1} \circ \Omega_q^{k_1} \quad X_m^{k_2} \circ \Omega_q^{k_2} \quad \cdots X_m^{k_r} \circ \Omega_q^{k_r} \vdots \omega_{qN+1}, \omega_{qN+2}\right]_{m=1(1)M} \\ &\text{with } \sum_{j=1}^{r} k_j = N \text{ and all } k_j,\ N,\ r,\ q \in N. \end{aligned} \tag{6}$$

Definition 3 The input matrix $\mathbf{X} = [x_{mn}]$ is transformed according to the combined values of the two elements of its core block $(\omega_{qN+1}, \omega_{qN+2})$ and its segments $\Omega^k_m; k \in \mathbf{N}$, into the evolutionary data matrix $\Xi^q = \left[\xi^q_{mk}\right], m = 1, 2, \ldots, M, k = 1, 2, \ldots, k_r$, with $\sum_{j=1}^{r} k_j = N$, and all k_j, N, r, $q \in \mathbf{N}$, as follows:

$$\Xi^q \leftarrow X \circ \Omega_q : \left[\xi^q_{mk}\right] = \begin{bmatrix} \xi^q_{11} & \xi^q_{12} & \cdots & \xi^q_{1k_r} \\ \xi^q_{21} & \xi^q_{22} & \cdots & \xi^q_{2k_r} \\ \vdots & \vdots & \vdots & \vdots \\ \xi^q_{M1} & \xi^q_{M2} & \cdots & \xi^q_{Mk_r} \end{bmatrix} \tag{7}$$

and this $\forall q = 1, 2, \ldots, Q$ with,

$$\xi^q_{mk} = \begin{cases} \sum_{k=1}^{k_r} x_{mk}\omega_{mk} & \text{if} \quad (\omega_{qN+1}, \omega_{qN+2}) = (0,0) \\ \sum_{k=1}^{k_r} x_{mk}\omega_{mk}/k_r & \text{if} \quad (\omega_{qN+1}, \omega_{qN+2}) = (1,1) \\ \text{Median of } \mathbf{X}^{k_r}_m & \text{if} \quad (\omega_{qN+1}, \omega_{qN+2}) = (0,1) \\ \text{Max } \mathbf{X}^{k_r}_m - \text{Min } \mathbf{X}^{k_r}_m & \text{if} \quad (\omega_{qN+1}, \omega_{qN+2}) = (1,0) \end{cases} \tag{8}$$

In other words, if the core genes are zero, then only the first element of each segment is returned. If they are ones then the average of each segment is returned. In the cases where the core genes are (0,1), then the median of each segment is returned and finally, if they are (1,0), then the maximum value minus the minimum value of each segment is returned, provided for both the last two cases that the numbers in each segment have been ordered appropriately.

Thus, we end up with a number of secondary data sets equal to the number of trainers, comprising the matrix $\Xi = [\Xi^q]_{q=1(1)Q}$. Each data set consists of an equal number of records as the initial time series, but reduced to size (i.e. width), according to the segments designed by each trainer:

$$\Xi = [\Xi^q]_{q=1(1)Q} \leftarrow \left[\mathbf{X} \circ \Omega_q\right]_{q=1(1)Q} = \left[\begin{bmatrix} \xi^1_{11} & \xi^1_{12} & \cdots & \xi^1_{1r_1} \\ \xi^1_{21} & \xi^1_{22} & \cdots & \xi^1_{2r_1} \\ \vdots & \vdots & \vdots & \vdots \\ \xi^1_{M1} & \xi^1_{M2} & \cdots & \xi^1_{Mr_1} \end{bmatrix} \begin{bmatrix} \xi^2_{11} & \xi^2_{12} & \cdots & \xi^2_{1r_2} \\ \xi^2_{21} & \xi^2_{22} & \cdots & \xi^2_{2r_2} \\ \vdots & \vdots & \vdots & \vdots \\ \xi^2_{M1} & \xi^2_{M2} & \cdots & \xi^2_{Mr_2} \end{bmatrix} \cdots \begin{bmatrix} \xi^Q_{11} & \xi^Q_{12} & \cdots & \xi^Q_{1r_M} \\ \xi^Q_{21} & \xi^Q_{22} & \cdots & \xi^Q_{2r_M} \\ \vdots & \vdots & \vdots & \vdots \\ \xi^Q_{M1} & \xi^Q_{M2} & \cdots & \xi^Q_{Mr_M} \end{bmatrix}\right] \tag{9}$$

The finally produced evolutionary data sets result after the output matrix $\mathbf{Y}$ has been appended to each Ξ^q row wise.

Working example Let us consider a classification dual class time series problem, where each of the two classes (*Class1* and *Class2*) corresponds to a 10-measurement set. The task is to correctly classify an unknown 10-measurement time series record. Supposedly, the initial raw data is given by the following table:

$$\begin{bmatrix} 3 & 8 & 7 & 8 & 2 & 1 & 4 & 5 & 9 & 6 & Class1 \\ 2 & 9 & 4 & 3 & 5 & 2 & 6 & 3 & 8 & 7 & Class1 \\ 7 & 6 & 3 & 9 & 2 & 1 & 5 & 4 & 6 & 8 & Class2 \\ 5 & 4 & 5 & 6 & 3 & 4 & 2 & 1 & 7 & 8 & Class1 \\ 6 & 3 & 2 & 4 & 7 & 5 & 8 & 9 & 1 & 4 & Class1 \\ 4 & 7 & 6 & 5 & 1 & 8 & 3 & 2 & 9 & 5 & Class2 \\ 9 & 5 & 8 & 1 & 4 & 6 & 7 & 8 & 4 & 2 & Class2 \\ 8 & 2 & 9 & 7 & 6 & 3 & 1 & 5 & 3 & 1 & Class2 \end{bmatrix}$$

By creating a binary output matrix for *Class1* $= 0$ and *Class2* $= 1$, we may break the initial raw data set into the input $\mathbf{X}$ and output $\mathbf{Y}$ matrices as follows:

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 3 & 8 & 7 & 8 & 2 & 1 & 4 & 5 & 9 & 6 \\ 2 & 9 & 4 & 3 & 5 & 2 & 6 & 3 & 8 & 7 \\ 7 & 6 & 3 & 9 & 2 & 1 & 5 & 4 & 6 & 8 \\ 5 & 4 & 5 & 6 & 3 & 4 & 2 & 1 & 7 & 8 \\ 6 & 3 & 2 & 4 & 7 & 5 & 8 & 9 & 1 & 4 \\ 4 & 7 & 6 & 5 & 1 & 8 & 3 & 2 & 9 & 5 \\ 9 & 5 & 8 & 1 & 4 & 6 & 7 & 8 & 4 & 2 \\ 8 & 2 & 9 & 7 & 6 & 3 & 1 & 5 & 3 & 1 \end{bmatrix} \text{ and}$$

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Suppose that the matrix of trainers $\boldsymbol{\Omega}$ consists of six records (trainers), each of which comprises of 12 (=10 + 2) elements, namely:

$$\boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{\Omega}_1 \\ \boldsymbol{\Omega}_2 \\ \boldsymbol{\Omega}_3 \\ \boldsymbol{\Omega}_4 \\ \boldsymbol{\Omega}_5 \\ \boldsymbol{\Omega}_6 \end{bmatrix} = \left[\begin{array}{cccccccccc:cc} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

Thus, each trainer comprises of the following segments

$$\boldsymbol{\Omega}_1 = \boldsymbol{\Omega}_1^3\boldsymbol{\Omega}_1^1\boldsymbol{\Omega}_1^3\boldsymbol{\Omega}_1^2\boldsymbol{\Omega}_1^1 \vdots 00 \qquad \boldsymbol{\Omega}_2 = \boldsymbol{\Omega}_2^1\boldsymbol{\Omega}_2^2\boldsymbol{\Omega}_2^4\boldsymbol{\Omega}_2^3 \vdots 11$$

$$\boldsymbol{\Omega}_3 = \boldsymbol{\Omega}_3^2\boldsymbol{\Omega}_3^3\boldsymbol{\Omega}_3^3\boldsymbol{\Omega}_3^1\boldsymbol{\Omega}_3^1 \vdots 10 \qquad \boldsymbol{\Omega}_4 = \boldsymbol{\Omega}_4^1\boldsymbol{\Omega}_4^4\boldsymbol{\Omega}_4^3\boldsymbol{\Omega}_4^1\boldsymbol{\Omega}_4^1 \vdots 01$$

$$\boldsymbol{\Omega}_5 = \boldsymbol{\Omega}_5^6\boldsymbol{\Omega}_5^1\boldsymbol{\Omega}_5^3 \vdots 11 \qquad \boldsymbol{\Omega}_6 = \boldsymbol{\Omega}_6^2\boldsymbol{\Omega}_6^1\boldsymbol{\Omega}_6^3\boldsymbol{\Omega}_6^4 \vdots 10$$

Each one of these trainers will produce an evolutionary data set by mapping its segmentation scheme on the initial raw time series data. As an illustration, the application of $\boldsymbol{\Omega}_1$ on the initial raw data set yields:

$$\boldsymbol{\Xi}^1 = \mathbf{X} \circ \boldsymbol{\Omega}_1 = \left[\mathbf{X}_m^3\,\mathbf{X}_m^1\mathbf{X}_m^3\,\mathbf{X}_m^2\,X_m^1\right]_{m=1(1)8} \circ \left[\boldsymbol{\Omega}_1^3\,\boldsymbol{\Omega}_1^1\,\boldsymbol{\Omega}_1^3\,\boldsymbol{\Omega}_1^2\,\boldsymbol{\Omega}_1^1\right]$$
$$= \left[\mathbf{X}_m^3 \circ \boldsymbol{\Omega}_1^3 \quad \mathbf{X}_m^1 \circ \boldsymbol{\Omega}_1^1 \quad \mathbf{X}_m^3 \circ \boldsymbol{\Omega}_1^3 \quad \mathbf{X}_m^2 \circ \boldsymbol{\Omega}_1^2 \quad \mathbf{X}_m^1 \circ \boldsymbol{\Omega}_1^1\right]_{m=1(1)8}$$

Thus

$$\boldsymbol{\Xi}^1 = \begin{bmatrix} 3 & 8 & 2 & 5 & 6 \\ 2 & 3 & 5 & 3 & 7 \\ 7 & 9 & 2 & 4 & 8 \\ 5 & 6 & 3 & 1 & 8 \\ 6 & 4 & 7 & 9 & 4 \\ 4 & 5 & 1 & 2 & 5 \\ 9 & 1 & 4 & 8 & 2 \\ 8 & 7 & 6 & 5 & 1 \end{bmatrix} \Rightarrow \text{EDS}_1 = \begin{bmatrix} 3 & 8 & 2 & 5 & 6 & 0 \\ 2 & 3 & 5 & 3 & 7 & 0 \\ 7 & 9 & 2 & 4 & 8 & 1 \\ 5 & 6 & 3 & 1 & 8 & 0 \\ 6 & 4 & 7 & 9 & 4 & 0 \\ 4 & 5 & 1 & 2 & 5 & 1 \\ 9 & 1 & 4 & 8 & 2 & 1 \\ 8 & 7 & 6 & 5 & 1 & 1 \end{bmatrix}$$

where the evolutionary data set EDS_1 results after the output matrix Y has been appended row wise. In the above, the operator '$\circ$' is defined as an element by element multiplication of the two arrays. This behaviour is dictated by the core block of the trainer (0,0).

In a similar fashion, the core block of the trainer $\boldsymbol{\Omega}_2$ is (1,1) and so the evolutionary data set will comprise of the average for each segment:

$$\boldsymbol{\Xi}^2 = \mathbf{X} \circ \boldsymbol{\Omega}_2 = \left[\mathbf{X}_m^1\,\mathbf{X}_m^2\,\mathbf{X}_m^4\,\mathbf{X}_m^3\right]_{m=1(1)8} \circ \left[\boldsymbol{\Omega}_2^1\,\boldsymbol{\Omega}_2^2\,\boldsymbol{\Omega}_2^4\,\boldsymbol{\Omega}_2^3\right]$$
$$= \left[\mathbf{X}_m^1 \circ \boldsymbol{\Omega}_2^1 \quad \mathbf{X}_m^2 \circ \boldsymbol{\Omega}_2^2 \quad \mathbf{X}_m^4 \circ \boldsymbol{\Omega}_2^4 \quad \mathbf{X}_m^3 \circ \boldsymbol{\Omega}_2^3\right]_{m=1(1)8}$$

Thus

$$\boldsymbol{\Xi}^2 = \begin{bmatrix} 3 & 7\frac{1}{2} & 3\frac{3}{4} & 6\frac{2}{3} \\ 2 & 6\frac{1}{2} & 4 & 6 \\ 7 & 4\frac{1}{2} & 4\frac{1}{4} & 6 \\ 5 & 4\frac{1}{2} & 3\frac{3}{4} & 5\frac{1}{3} \\ 6 & 2\frac{1}{2} & 6 & 4\frac{2}{3} \\ 4 & 6\frac{1}{2} & 4\frac{1}{4} & 5\frac{1}{3} \\ 9 & 6\frac{1}{2} & 4\frac{2}{4} & 4\frac{2}{3} \\ 8 & 3 & 4\frac{1}{4} & 3 \end{bmatrix} \Rightarrow \text{EDS}_2 = \begin{bmatrix} 3 & 7\frac{1}{2} & 3\frac{3}{4} & 6\frac{2}{3} & 0 \\ 2 & 6\frac{1}{2} & 4 & 6 & 0 \\ 7 & 4\frac{1}{2} & 4\frac{1}{4} & 6 & 1 \\ 5 & 4\frac{1}{2} & 3\frac{3}{4} & 5\frac{1}{3} & 0 \\ 6 & 2\frac{1}{2} & 6 & 4\frac{2}{3} & 0 \\ 4 & 6\frac{1}{2} & 4\frac{1}{4} & 5\frac{1}{3} & 1 \\ 9 & 6\frac{1}{2} & 4\frac{2}{4} & 4\frac{2}{3} & 1 \\ 8 & 3 & 4\frac{1}{4} & 3 & 1 \end{bmatrix}$$

where, the second evolutionary data set EDS_2 results again after the output matrix Y has been appended row wise.

As it appears, there will be six evolutionary data sets in total, since the trainer matrix consists of equal number of trainers, which are all produced likewise. For the sake of completeness of the working example, we provide the EDS_3 and EDS_4 for which the behaviour is dictated by the core block of the trainer (0,1) and (1,0), respectively. Therefore,

$$\boldsymbol{\Xi}^3 = \mathbf{X} \circ \boldsymbol{\Omega}_3 = \left[\mathbf{X}_m^2\,\mathbf{X}_m^3\mathbf{X}_m^3\,\mathbf{X}_m^1\,\mathbf{X}_m^1\right]_{m=1(1)8} \circ \left[\boldsymbol{\Omega}_3^2\boldsymbol{\Omega}_3^3\boldsymbol{\Omega}_3^3\boldsymbol{\Omega}_3^1\boldsymbol{\Omega}_3^1\right]$$
$$= \left[\mathbf{X}_m^2 \circ \boldsymbol{\Omega}_3^2 \quad \mathbf{X}_m^3 \circ \boldsymbol{\Omega}_3^3 \quad \mathbf{X}_m^3 \circ \boldsymbol{\Omega}_3^3 \quad \mathbf{X}_m^1 \circ \boldsymbol{\Omega}_3^1 \quad \mathbf{X}_m^1 \circ \boldsymbol{\Omega}_3^1\right]_{m=1(1)8}$$

Thus

$$\boldsymbol{\Xi}^3 = \begin{bmatrix} 5\frac{1}{2} & 7 & 4 & 9 & 6 \\ 5\frac{1}{2} & 4 & 3 & 8 & 7 \\ 6\frac{1}{2} & 3 & 4 & 6 & 8 \\ 4\frac{1}{2} & 5 & 2 & 7 & 8 \\ 4\frac{1}{2} & 4 & 8 & 1 & 4 \\ 5\frac{1}{2} & 5 & 3 & 9 & 5 \\ 7 & 4 & 7 & 4 & 2 \\ 5 & 7 & 5 & 3 & 1 \end{bmatrix} \Rightarrow \text{EDS}_3 = \begin{bmatrix} 5\frac{1}{2} & 7 & 4 & 9 & 6 & 0 \\ 5\frac{1}{2} & 4 & 3 & 8 & 7 & 0 \\ 6\frac{1}{2} & 3 & 4 & 6 & 8 & 1 \\ 4\frac{1}{2} & 5 & 2 & 7 & 8 & 0 \\ 4\frac{1}{2} & 4 & 8 & 1 & 4 & 0 \\ 5\frac{1}{2} & 5 & 3 & 9 & 5 & 1 \\ 7 & 4 & 7 & 4 & 2 & 1 \\ 5 & 7 & 5 & 3 & 1 & 1 \end{bmatrix}$$

Finally,

$$\boldsymbol{\Xi}^4 = \mathbf{X} \circ \boldsymbol{\Omega}_4 = \left[\mathbf{X}_m^1\,\mathbf{X}_m^4\mathbf{X}_m^3\,\mathbf{X}_m^1\,\mathbf{X}_m^1\right]_{m=1(1)8} \circ \left[\boldsymbol{\Omega}_4^1\boldsymbol{\Omega}_4^4\boldsymbol{\Omega}_4^3\boldsymbol{\Omega}_4^1\boldsymbol{\Omega}_4^1\right]$$
$$= \left[\mathbf{X}_m^1 \circ \boldsymbol{\Omega}_4^1 \quad \mathbf{X}_m^4 \circ \boldsymbol{\Omega}_4^4 \quad \mathbf{X}_m^3 \circ \boldsymbol{\Omega}_4^3 \quad \mathbf{X}_m^1 \circ \boldsymbol{\Omega}_4^1 \quad \mathbf{X}_m^1 \circ \boldsymbol{\Omega}_4^1\right]_{m=1(1)8}$$

Thus

$$\boldsymbol{\Xi}^4 = \begin{bmatrix} 0 & 6 & 4 & 0 & 0 \\ 0 & 6 & 4 & 0 & 0 \\ 0 & 7 & 4 & 0 & 0 \\ 0 & 3 & 3 & 0 & 0 \\ 0 & 5 & 4 & 0 & 0 \\ 0 & 6 & 6 & 0 & 0 \\ 0 & 7 & 2 & 0 & 0 \\ 0 & 7 & 4 & 0 & 0 \end{bmatrix} \Rightarrow \text{EDS}_4 = \begin{bmatrix} 0 & 6 & 4 & 0 & 0 & 0 \\ 0 & 6 & 4 & 0 & 0 & 0 \\ 0 & 7 & 4 & 0 & 0 & 1 \\ 0 & 3 & 3 & 0 & 0 & 0 \\ 0 & 5 & 4 & 0 & 0 & 0 \\ 0 & 6 & 6 & 0 & 0 & 1 \\ 0 & 7 & 2 & 0 & 0 & 1 \\ 0 & 7 & 4 & 0 & 0 & 1 \end{bmatrix}$$

3 System mechanics

The developed tool essentially designs the best possible segmentation scheme on the initial raw time series data, by monitoring the performance of a certain number of generations of trainers, while also deciding on the best classifier (the ANN or the SVM) to use for the certain problem each time (Fig. 1). Upon initialization, the system reads the raw time series data set, creates the training and testing sets and from then on, proceeds by generations. In our case, the stopping condition of the algorithm is the number of generations (1,000). Inside each generation, the first step of the system is to compile a pool of trainers to manipulate the training and testing data sets. If this is the first generation, the trainers are chosen at random, whereas if it is a subsequent generation, the pool consists of trainers of the previous generation, chosen according to the 'roulette wheel' selection policy. Table 2 presents a pseudo code sample of the evolutionary data production process. The interested reader may ask for the full source code of the method, which is freely available.

After the training and testing phase, each trainer is assigned a fitness score which is derived by the trainer's performance assessed by the accuracy of the classifier (ANN/SVM) object. This is essentially the proximity of the potential of the trainer to an optimal solution set in the initialization of the process. Let r_{ij} be the accuracy value of the classifier trained and tested with the evolutionary data produced by the ith trainer of the jth population of the algorithm. Then, the fitness score f_{ij} assigned to the ith trainer of the jth population should be $f_{ij} = \frac{1}{|r_{ij}-h|}$, where h is the accuracy threshold, which maximizes the f_{ij}. As f_{ij} rises in value, r_{ij} moves closer to h and our system crawls nearer to the ideal solution h arbitrarily set in the beginning of the algorithm. Maximizing f_{ij} produces stronger and more potent populations of trainers. Note that the accuracy of the classifier object is measured during the testing phase, where it is applied on totally unseen cases, that is, the evaluation of evolutionary data set.

The algorithm then updates each trainer with the corresponding fitness score. According to the roulette wheel selection object, with which the next generation genetic pool is compiled, the fitness score f_{ij} of the ith trainer ($i = 1,2,\ldots,V$) of the jth population ($j = 1,2,\ldots,M$) has probability $p_{ij} = f_{ij} / \sum_{i=1}^{V} f_{ij}, \quad \forall j = 1, 2, \ldots, M$ of being selected.

Following this procedure, the frequency of each trainer in the next generation genetic pool is proportionate to its potential. After the fitness score has been assigned to each trainer, the system selects the fitter ones, perpetuating the fittest trainers, that is, the ones with the higher fitness score. In this context, trainers with relatively high fitness scores are less likely to be eliminated. On the other hand, less fit trainers may not be extinguished from the genetic pool of the next generations. This results in the fact that some weaker solutions to the problem at hand may survive the algorithm sweep for the forming of the next generations, conveying their potentially useful genes to their offspring. The genetic pool is the interim between successive generations where mating happens and the next generation of trainers is produced. Here, the offspring may mate and give birth to the same (selection) or combined (crossover) trainer, the genes of which may undergo mutation with a certain probability. The process of mating continues until the number of the next generation trainers is completed. In the end, the system elects the best trainer, the best evolutionary segmentation scheme, as well as the best classifier, trained and ready for use on unseen cases.

4 Use cases

The proposed methodology was applied for the solution of two problems. The first case is essentially a classification problem [2]. In this research, the bioelectric recognition assay (BERA) method [48, 49] was engaged so as to provide information used in the detection and identification of certain plant viruses, namely the *tobacco rattle virus* (TRV) and the *cucumber green mottle mosaic virus* (CGMMV), using appropriately preprocessed reagents as the sensing elements. While reacting to the biosensors, each of the viruses in question exhibit unique patterns of biosensor responses over specific ranges of concentrations, rendering these responses as a special characteristic for each virus, a real identification signature. Each signature is in essence a graphical curve of bioelectrical responses in the time unit, a time series data set, which should be identified as a characteristic for each virus and effectively classified.

The second problem on which PES was applied relates to the management of water reservoirs [1]. The study area was Cyprus and the main motivation was that despite the fact that time consuming studies had been conducted during the previous years, not much had been achieved towards the development of a viable solution to the Cypriot water resources management problem. The inputs of the problem include structural and dynamic data, in which monthly precipitation particles play a distinct role. In this case, the time series information originated from the historical monthly rainfall data measured at certain watershed stations for a wide temporal period. The issue here was to develop a methodology for the production of evolutionary training/testing data, in order to achieve an effective estimation of the average annual water supply (AAWS) on an annual basis, for each mountainous

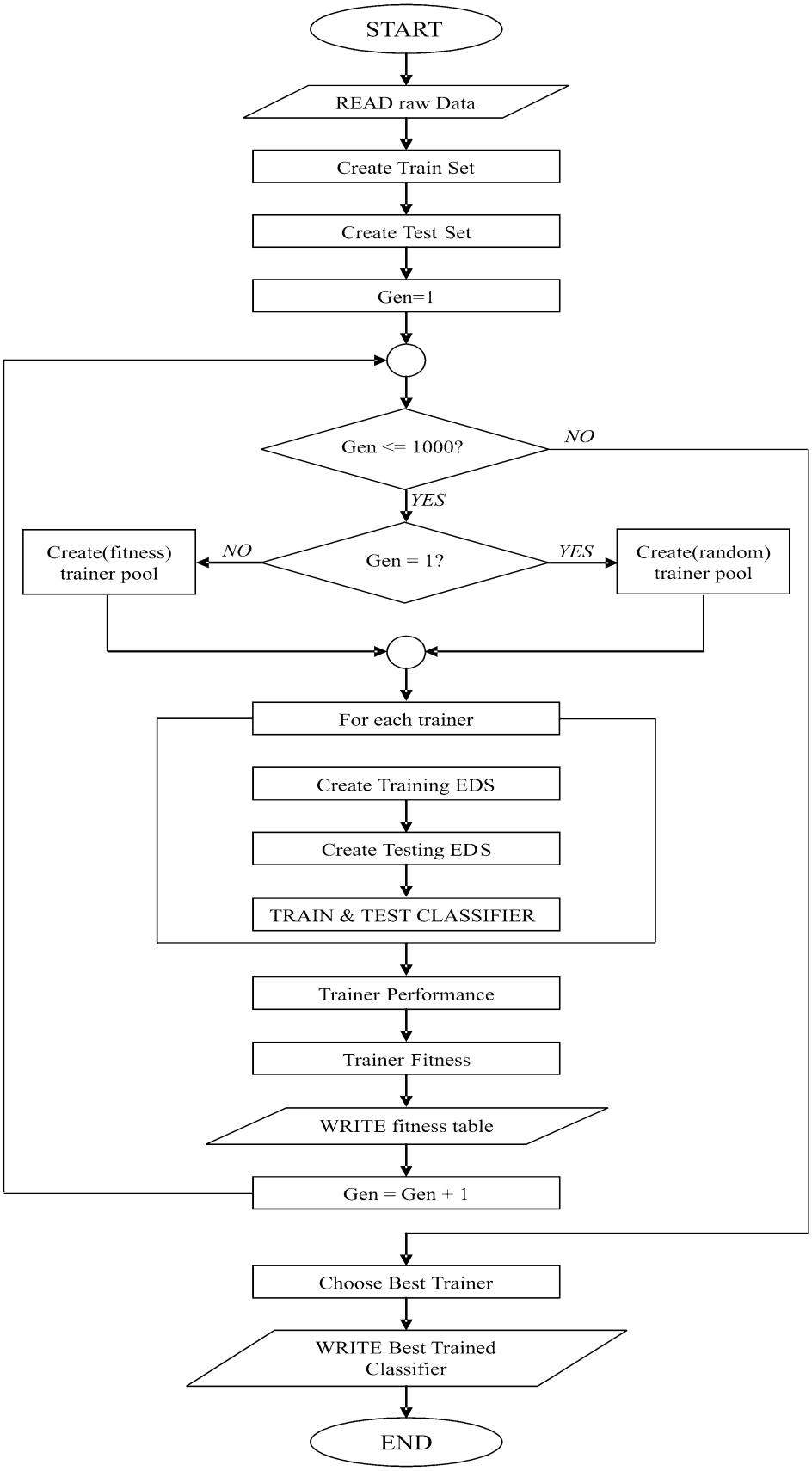


Fig. 1 Flowchart of system function

Table 2 EDS production pseudocode

```
METHOD createEDS_Data(generation, crsvrProbability, mtnProbability):
        READ initial time series data IN tsDat
        numTrnrs = number of Trainers per Generation
        CREATE_EMPTY_ARRAY newTrnrPop              # Population of next generation
        IF generation == 1 THEN
          WHILE i <= numTrnrs
            POPULATE newTrnrPop with Trainer(i)    # Each trainer must be tsDat wide
            i = i+1
          END WHILE
        ELSE
          WHILE i <= numTrnrs
            trnr1,trnr2 = Choose two trainers randomly from newTrnrPop
            crsvrPosition = Randomly define the position for crossover
            crsvrActivate = Random integer in [0,100]
            IF crsvrActivate < crsvrProbability THEN    # Decide if crossover fires
              newTrnrPop.append(trnr1[0:crsvrPosition]+trnr2[crsvrPosition:end])
              newTrnrPop.append(trnr2[0:crsvrPosition]+trnr1[crsvrPosition:end])
            ELSE
              TRANSFER (trnr1,trnr2) to newTrnrPop      # If crossover fails choose parents
            END IF
            mtnActivate= Random integer in [0,1000]
            FOR each individual Trainer
              FOR each individual gene
                IF mtnActivate < mtnProbability THEN    # Decide if mutation fires
                  Change gene from 0 to 1 and vice - versa
                END IF
              END FOR
            END FOR
            i = i+1
          END WHILE
        END IF
        FOR trainer IN newTrnrPop                   # Segmentation scheme mapping
          IF (trainer[0],trainer[1]) == (0, 0) THEN     # trainer[0],trainer[1]: core genes
            EDS_data = Discard_Zeros(trainer[2:],tsDat)     #
          END IF
          IF (trainer[0],trainer[1]) == (1, 1) THEN     # FOLZ: First One Last Zero
            EDS_data = FOLZ_average(trainer[2:],tsDat)    # Segment approximation: average
          END IF
          IF (trainer[0],trainer[1]) == (0, 1) THEN
            EDS_data = FOLZ_median(trainer[2:],tsDat)     # Segment approximation: median
          END IF
          IF (trainer[0],trainer[1]) == (1, 0) THEN
            EDS_data = FOLZ_minmax(trainer[2:],tsDat)     # Segment approximation: max-min
          END IF
        RETURN EDS_data
```

Table 3 System parameterization

Sub-routine	Use case	ANN	SVM
Classifier type	Virus Id	Multi-layer perceptron	C-SVC
	Torrents		C-SVR
Number of hidden neurons	Virus Id	*((In + Out)/2) + 0.1*(records)*	
	Torrents	*Int(log(records)*log(inputs))*	
Activation function hidden	Both	Sigmoid	
Activation function output	Virus Id	SoftMax	
	Torrents	Linear	
SVM Kernel	Both		RBF
C,γ parameters	Both		Grid search
Normalization	Both	[−1, 1]	[0, 1]
Trainer population	Both	15	
Generations	Both	1,000	
Selection method	Both	Roulette wheel	
Crossover rate	Both	40 %	
Mutation rate	Both	5 ‰	

watershed of Cyprus. The estimation of the aforementioned factor is crucial and plays a highly important role to the management of mountainous water reservoirs, as it is closely related to the mountainous watershed fermentations, as well as to the potential torrential risks posed on the areas involved.

The proposed system was developed under Ubuntu Linux and constitutes a cross-platform application. Python v.2.7 was used as programming language, enhanced with the PyBrain v.0.3 library for the ANN object and the Scikits.Learn v.0.8.1 library for the SVM object. The genetic algorithm was developed from scratch. The software tool was tested on a PC system utilizing an Intel Core i7 CPU processor at 3.07 GHz with 6 GB of RAM, with system parameterization as presented in Table 3.

The proposed system is designed such that the fittest SVM parameters C and γ are calculated for each evolutionary data set by utilizing a grid-search procedure under a fivefold cross-validation scheme. This sub-routine precedes the actual training phase and separates each evolutionary training set into 5 equally sized subsets. For each iteration of the procedure, the system is trained with the fourfolds keeping one for the testing. The training of the system, along with the subsequent testing aims to reveal the right combination of the SVM parameters C and γ. The former is derived by an array of numbers from 250 to 1,000, while the latter comes from an array of powers of 2 [50]. The sub-routine returns the average of the best combinations of the two parameters in the fivefold cross-validation testing for each evolutionary training set. The same combination of C and γ is kept for the testing phase.

The overall training times for each case study and classifier are given in Table 4.

Table 4 Training times for both use cases

	ANN		SVM	
	Torrential risk	Virus Id.	Torrential risk	Virus Id.
Training time				
Days	6	8	0	3
Hours	23	15	16	7
Mins.	28	41	48	27
Secs.	6	44	26	11

It might be noted that, although SVM training is faster than all other cases examined, it generally took a comparatively long time for the training phase to complete (Table 4). This might be justified by the interpreted nature of the development language to a great extent. Another crucial reason lies in the mass of the produced evolutionary representations of the original data, combined with the system parameterization routines. Indeed, during the course of the evolutionary procedure, 15,000 such data sets per problem were generated, for each of which, SVM parameter optimization and ANN hidden neurons customization preceded the actual training and testing phases.

4.1 Classification: plant virus identification case study

In this case, BERA method was engaged in order to detect certain plant viruses, especially the TRV and the CGMMV. According to the method proposed in [48, 49], appropriately preprocessed reagents are used as biosensor elements in a reaction towards each of the viruses in a number of iterations. While reacting to these specially constructed biosensors, each virus exhibits unique patterns of electrochemical responses which are monitored over specific

ranges of concentrations and quantified as voltage differences in the time unit. The produced time series data stand as a characteristic signature for each virus and is utilized in the classification procedure.

The data set acquired by the BERA sensors consisted of 1,271 records, each of which was a time series of 331 time fragments. The initial information was split in the training and testing data set consisting of 1,000 and 192 instances, respectively, while 79 records were set aside to form the evaluation data set.

The classification results were derived after training the classifier with the training data set and then running the trained classifier on the corresponding testing data set. The *accuracy* metric of each trainer is derived as the ratio of the sum of true positive (TP) and true negative (TN) instances by the sum of the total instances in the testing data set. Thus,

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{10}$$

In our case, TPs are the instances which truly are CGMMV and are classified as such, false positives (FP) are those that are CGMMV, but are classified as TRV. Similarly, TNs are the instances that are TRV and are correctly classified, while false negatives (FN) are those that are TRV, but are classified as CGMMV. The performance of the trainer, that is, its classification potential, is also measured by the sensitivity (Sens), the specificity (Spec), the positive prediction value (PPV) and its negative prediction value (NPV), metrics also offered by the tool for each trainer. In our case, the *sensitivity* and *specificity* metrics quantify the likelihood of the tool itself to correctly classify the records. *Sensitivity* measures the proportion of the CGMMV instances which are correctly identified as such, while *specificity* stands for the TRV proportion. PPV and NPV measure the likelihood a virus identified as a CGMMV or TRV, respectively, to actually be one. Thus,

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad \text{Specificity} = \frac{\text{TN}}{\text{FP} + \text{TN}} \tag{11}$$

$$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{NPV} = \frac{\text{TN}}{\text{TN} + \text{FN}} \tag{12}$$

Results for the classification of the initial raw unchanged time series data set, as well as the five best performing trainers from the evolutionary procedure, are shown in Table 5 and Fig. 2.

4.2 Regression: torrential risk case study

The research area covers all of the mountainous watersheds under the administration of the Republic of Cyprus. The initial information was accumulated by 78 stations located at the span of 70 torrential streams, covering a temporal period of 28 years, from 1965 to 1993, for most of the stations' measurements. The input data, according to their type, have been split into two categories: structural data remain constant throughout the period of measurement, while dynamic data encompass ever changing variables.

All in all, 1,273 patterns of data were accumulated, split into 1,100 train, 100 test and 73 evaluation data sets, respectively. The inputs to the system were the area of the watershed, the absolute altitude and the absolute slope as structural input data, whereas the time series portion of the

Table 5 Results of classification: raw time series and evolutionary data in ANN and SVM training, sorted by *accuracy*

Generation	TP	FP	FN	TN	Sens	Spec	PPV	NPV	Acc	Neurons
Artificial neural network										
150	74	17	7	94	0.813	0.847	0.813	0.931	0.875	193
155	67	15	14	96	0.817	0.865	0.817	0.873	0.849	194
826	72	21	9	90	0.774	0.811	0.774	0.909	0.844	190
287	71	22	10	89	0.763	0.802	0.763	0.899	0.833	182
826	76	28	5	83	0.731	0.748	0.731	0.943	0.828	190
Raw TS	72	84	9	27	0.889	0.243	0.462	0.750	0.516	182
Generation	TP	FP	FN	TN	Sens	Spec	PPV	NPV	Acc	C	γ
Support vector machine											
209	80	5	1	106	0.941	0.955	0.941	0.991	0.969	756.5	11.2
662	80	6	1	105	0.930	0.946	0.930	0.991	0.963	757.1	7.5
773	79	6	2	105	0.929	0.946	0.929	0.981	0.958	756.1	46.2
999	78	6	3	105	0.929	0.946	0.929	0.972	0.953	756.8	53.1
998	78	7	3	104	0.918	0.937	0.918	0.972	0.948	754.4	49.4
Raw TS	67	8	14	103	0.827	0.928	0.893	0.880	0.885	786.8	37.4

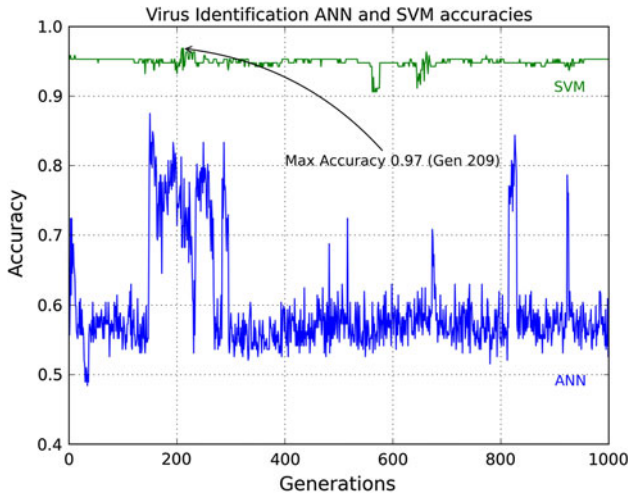


Fig. 2 ANN and SVM accuracies for the virus identification classification problem

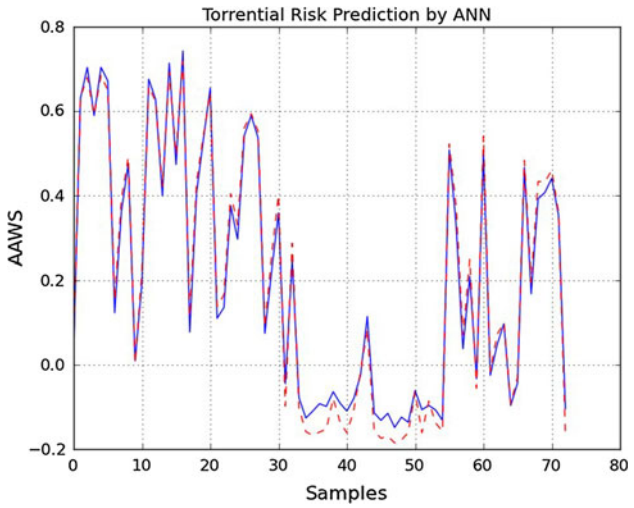


Fig. 3 Torrential risk prediction on the evaluation set, using ANN trained by the best achieved piecewise evolutionary segmentation of the 28th generation. The *solid line* represents system prediction, while the actual values are depicted by the *dotted line*

input data consisted of the maximum water supply, the average annual rainfall, as well as the average rainfall for each month, for each year of measurement. The network had only one output, the AAWS. Each trainer's fitness was calculated as a function of the root mean square error (RMSE) during the testing phase of the system. The RMSE metric exhibits the quality of model representation, that is, lower RMSE shows better model approximation. Derived results are given in Table 6 and Fig. 3, where it is shown that PES has achieved to enhance the power of both classifiers.

Examining the results derived by the two use cases more closely and comparing the performance of the classifiers

Table 6 Results of regression: raw time series and evolutionary data in ANN and SVM training, sorted by RMSE ascending

ANN			SVM			
Generation	RMSE	Neurons	Generation	RMSE	C	γ
28	59e−4	14	1	64e−3	1,000	8
49	62e−4	15	4	65e−3	1,000	8
90	63e−4	16	394	66e−3	1,000	4
31	72e − 4	13	329	67e−3	400	4
29	76e−4	14	1	68e−3	700	8
Raw TS	64e−3	19	Raw TS	68e−3	1,000	4

with and without the evolutionary segmentation method, we come to the following conclusions:

- PES time series model analysis eventually provides a potent representation scheme for both problems, effectively approximating initial time series data in spaces of lower dimensionality. The extraction of significant characteristics enables the system to combine preprocessed data and computational intelligence classifiers in an integrated tool which adapts to the data explored
- PES has enhanced the predictive and classification potential of all classifiers in every instance (Tables 5, 6). This is made emphatic in the case of the virus identification problem, where the time series is large and the representation very effective. The classifier potential optimization decreases with the width of the time series. This leads us to the remark that PES needs a large enough time series to create credible segments. Nevertheless, it has succeeded in electing the most fit time series representation, upgrading the classification and regression capabilities of both classifiers for the two experiments.
- Strong enhancement of the classification potential can be noted for the case of virus identification where we have a leap from 51.6 % accuracy to 87.5 % for the ANN, while the SVM accuracy is enhanced from 88.5 to 96.9 % for the same problem. The methodology elects the SVM as the best classification module for the virus identification problem.
- As regards to the torrential risk problem (regression), the performance enhancement, that is, the decrease in the RMSE, is not that impressive as in the former case, although it exists in a smaller degree. Nevertheless, when we train the winning classifier with the best fit trainer and then test the unseen data set, we may note that our system predicts correctly all the cases where there is a real threat for torrential risk, while relaxing in every other case (Fig. 3). For this problem, our system promotes the use of the ANN as the predictive module.

5 Conclusions

One of the most common techniques for approximating time series data is perhaps the group of methods collectively referred to as segmentation algorithms, which divide the given data into a series of segments and cater for their approximation with a given basic function. Specifically, the piecewise linear representation family of algorithms utilize various piecewise linear models as its approximating functions, which may be classified as regards to their data feed type and their approximation type. The former classifies them into on- and offline algorithms, while the latter into regression and interpolation ones. Online segmentation performs the approximation on the reception of new data, while offline requires the whole time series package beforehand.

In this paper, we present the design, development, implementation and testing of PES, an innovative PLR-related method, aiming to enhance classification and regression capabilities for time series data. The most significant contribution of the present work is an effective time series representation method integrated with two classifiers through a software tool. The proposed system focuses on preprocessing time series data to effectively define an optimum, dynamic, auto-adaptive training factor. Thus, piecewise data segmentation schemes are designed on the input data vector of the initial raw information in an evolutionary fashion and are tested as to their performance. The fittest evolutionary data set consists of the most substantial features of the initial information and constitutes a training scheme utilized to introduce vast enhancements to the classification or predictive potential of the classifier. The proposed method was tested against a regression and a classification problem, giving very promising results, based on the assumption that, under PES, there will eventually be a trainer which best describes the problem and enables the chosen classifier to be optimally trained. Indeed, as shown in Tables 5 and 6, PES has achieved to produce such trainers that effectively smooth out the input data in both cases and enhance the training of the classifiers, compared with the results yielded by the initial raw time series. Recapitulating the advantages, the following could be noted:

- The system presented in this paper has been enriched with two classifiers, that is, ANN and SVM, automatically electing the best one combined with the fittest evolutionary training data set for each problem. Thus, it compares two alternatives at the same time, while keeping the operating costs at acceptable levels.
- The proposed technique fits quite well with highly noisy or dimensional data in which each record consists of a large number of time series elements and constitutes a 'signature' for a certain class. Such is the case of research conducted towards identification of, say, either plant (like the case examined in this paper) or human viruses, or even pesticide residuals, the initial data of which stem from sensor experiments. Thus, the software tool might be of service in agricultural, biological or medical research, which often deals with time series problems of large width of data elements. Costly repetitions of the same experiment may also be significantly reduced by the vast number of PES generated possible representations of the initial information.
- Successful application of the proposed solution requires careful parameterization as regards to the genetic algorithm, as well as the classifier parameters. Associated with the nature of the initial time series information, the analysis must ascertain that the best number of hidden neurons for the ANN, as well as the optimum C and γ parameters for the SVM have been selected each time, thus internal grid-search routines have been embedded in the proposed solution for optimal parameterization.
- Taking into account that the algorithm has been designed to run under heavy conditions with optimal training and parameterization, it is natural to expect longer training times. Once trained though, the computational cost demands are reduced as PES, except from effectively smoothing out the input data, also succeeds in drastically reducing the total number of inputs in every case. This is essential in the economy of training because, once the fittest trainer has been revealed, the system transforms each new unseen case according to the segmentation scheme dictated by this trainer and runs only once. Furthermore, a fit segmentation and classifier combination are in fact proposed, which may then be embedded in any other software tool or information system to circumvent restraints posed by the development language.

Future research directions based on the proposed PES methodology introduced in this paper include tasks to enhance the overall optimization potential of the system, to incorporate procedures for alleviating the computational needs and to make it user friendlier. In the first category, we have planned to include more classifiers in the comparison scheme, such as the k-nearest neighbour (k-NN) or the Bayesian and decision tree ones. Also, prospects which pose definite targets for future research in this branch include embedding yet more advanced statistical metrics in the evolutionary algorithm or appointing the segments to enhanced filtering functions. Of course, changes such as these may burden the system with yet more computational power demands, thus a shift to genetic representations other than the binary might assist in this direction. Finally,

the design of a graphical user interface, which may encourage the acceptance of the proposed tool in other disciplines, is currently under construction.

References

1. Glezakos TJ, Tsiligiridis TA, Yialouris CP, Maris F, Ferentinos KP (2009) Feature extraction for time series data: an artificial neural network evolutionary training model for the management of mountainous watersheds. Neurocomputing 73(1–3):49–59
2. Glezakos TJ, Moschopoulou G, Tsiligiridis TA, Kintzios S, Yialouris CP (2010) Plant virus identification based on neural networks with evolutionary preprocessing. Comput Electron Agr 70(2):263–275
3. Haykin S (2008) Neural networks and learning machines. Prentice Hall, Englewood Cliffs
4. Boser BE, Guyon IM, Vapnik VN (1992) A training algorithm for optimal margin classifiers. In: Hausler D (ed) Proceedings of the fifth annual workshop on computational learning theory. ACM Press, Pittsburgh, pp 144–152
5. Chang CL, Lo SL, Yu SL (2005) Applying fuzzy theory and genetic algorithm to interpolate precipitation. J Hydrol 314:92–104
6. Chau KW (2007) A split-step particle swarm optimization algorithm in river stage forecasting. J Hydrol 346:131–135
7. Ni JR, Xue A (2003) Application of artificial neural network to the rapid feedback of potential ecological risk in flood diversion zone. Eng Appl Artif Intell 16:105–119
8. Recknagel F (2001) Applications of machine learning to ecological modelling. Ecol Model 146:303–310
9. Abraham A (2004) Meta learning evolutionary artificial neural networks. Neurocomputing 56:1–38
10. Elizondo DA, Birkenhead R, Gongora M, Taillard E, Luyima P (2007) Analysis and test of efficient methods for building recursive deterministic perceptron neural networks. Neural Netw 20:1095–1108
11. Niska H, Hiltunen T, Karppinen A, Ruuskanen J, Kolehmainen M (2004) Evolving the neural network model for forecasting air pollution time series. Eng Appl Artif Intell 17:159–167
12. Prudencio RBC, Ludermir TB (2004) Meta-learning approaches to selecting time series models. Neurocomputing 61:121–137
13. Rossi F, Delannay N, Conan-Guez B, Verleysen M (2005) Representation of functional data in neural networks. Neurocomputing 64:183–210
14. Sivagaminathan RK, Ramakrishnan S (2007) A hybrid approach for feature subset selection using neural networks and ant colony optimization. Expert Syst Appl 33:49–60
15. Diaz-Robles LA, Ortega JC, Fu JS, Reed GD, Chow JC, Watson JG, Moncada-Herrera JA (2008) A hybrid ARIMA and artificial neural networks model to forecast particulate matter in urban areas: the case of Temuco, Chile. Atmos Environ 42:8331–8340
16. Xiao Z, Ye S-J, Zhong B, Sun C-X (2009) BP neural network with rough set for short term load forecasting. Expert Syst Appl 36:273–279
17. Hamzacebi C (2008) Improving artificial neural networks' performance in seasonal time series forecasting. Inform Sci 178:4550–4559
18. Lu WZ, Wang WJ (2005) Potential assessment of the "support vector machine" method in forecasting ambient air pollutant trends. Chemosphere 59:693–701
19. Huang S-C, Wu T-K (2008) Integrating GA-based time-scale feature extractions with SVMs for stock index forecasting. Expert Syst Appl 35:2080–2088
20. Cao SG, Liu Y-B, Wang Y-P (2008) A forecasting and forewarning model for methane hazard in working face of coal mine based on LS-SVM. J China Univ Min Technol 18:0172–0176
21. Sun J, Zheng C, Zhou Y, Bai Y, Luo J (2008) Nonlinear noise reduction of chaotic time series based on multidimensional recurrent LS-SVM. Neurocomputing 71:3675–3679
22. Laskaris NA, Zafeiriou SP, Garefa L (2009) Use of random time-intervals (RTIs) generation for biometric verification. Pattern Recogn. doi:10.1016/j.patcog.2008.12.028
23. Coulibaly P, Evora ND (2007) Comparison of neural network methods for infilling missing daily weather records. J Hydrol 341:27–41
24. Du H, Zhang N (2008) Time series prediction using evolving radial basis function networks with new encoding scheme. Neurocomputing 71:1388–1400
25. Hung J-C (2008) A genetic algorithm approach to the spectral estimation of time series with noise and missed observations. Inform Sci. doi:10.1016/j.ins.2008.08.018
26. Cao H, Recknagel F, Joo GJ, Kim DK (2006) Discovery of predictive rule sets for chlorophyll-a dynamics in the Nakdong River (Korea) by means of the hybrid evolutionary algorithm HEA. Ecol Inf 1:43–53
27. Coelho JP, Moura Oliveira PB, Boaventura Cunha J (2005) Greenhouse air temperature predictive control using the particle swarm optimisation algorithm. Comput Electron Agr 49:330–344
28. Gaur S, Deo MC (2008) Real-time wave forecasting using genetic programming. Ocean Eng 35(11–12):1166–1172
29. Bodri L, Cermak V (2000) Prediction of extreme precipitation using a neural network: application to summer flood occurrence in Moravia. Adv Eng Softw 31:311–321
30. Hansen JV, McDonald JB, Nelson RD (1999) Time series prediction with genetic algorithm designed neural networks: an empirical comparison with modern statistical models. Comput Intell 15(3):171–184
31. Fu T (2011) A review on time series data mining. Eng Appl Artif Intel 24:164–181
32. Box G, Jenkins G (1976) Time series analysis: forecasting and control, revised edition. Holden-Day, Oakland
33. Velleman PW, Hoaglin DC (1981) Applications, basics, and computing of exploratory data analysis. Duxbury Press, Boston
34. Frank RJ, Davey N, Hunt SP (2001) Time series prediction and neural networks. J Intell Robot Syst 31(1–3):91–103
35. Rajagopalan V, Ray A, Samsi R, Mayer J (2007) Pattern identification in dynamical systems via symbolic time series analysis. Pattern Recogn 40(11):2897–2907
36. Lesher S, Guan L, Cohen AH (2000) Symbolic time-series analysis of neural data. Neurocomputing 32–33:1073–1081
37. Sharma KK, Joshi SD (2007) Time delay estimation using fractional Fourier transform. Signal Process 87:853–865
38. Giampaoli I, Ng WL, Constantinou N (2009) Analysis of ultra-high-frequency financial data using advanced Fourier transforms. Financ Res Lett 6:47–53
39. Bali TG (2008) The intertemporal relation between expected returns and risk. J Financ Econ 87:101–131
40. Bollerslev T (1986) Generalized autoregressive conditional heteroscedasticity. J Econom 31:307–327
41. Yamaguchi K (2008) Reexamination of stock price reaction to environmental performance: a GARCH application. Ecol Econ 68:345–352
42. Keogh E, Chu S, Hart D, Pazzani M (2004) Segmenting time series: a survey and novel approach. In: Last M, Kandel A, Bunke H (eds) Data mining in time series databases. World Scientific Pub Co Inc, Singapore, pp 1–21
43. Ding Y, Yang X, Kavs A, Li J (2010) A novel piecewise linear segmentation for time series. The 2nd international conference on computer and automation engineering (ICCAE), vol 4, pp 52–55,

Coll. of Comput. Sci. & Technol., Zhejiang Univ., Hangzhou, China
44. Guerrero J, Berlanga A, Garcia J, Molina M (2010) Piecewise linear representation segmentation as a multiobjective optimization problem. Distributed computing and artificial intelligence, AISC 79. Springer, Berlin, pp 267–274
45. Tseng VS, Chen CH, Huang PC, Hong TP (2009) Cluster-based genetic segmentation of time series with DWT. Pattern Recogn Lett 30:1190–1197
46. Wang XY, Wang ZO (2004) A structure-adaptive piece-wise linear segments representation for time series In: Zhang D, Gregoire E, DeGroot D (eds) Proceedings of the 2004 IEEE international conference on information reuse and integration, IRI: IEEE systems, man, and cybernetics society, pp 433–437
47. Glezakos TJ, Tsiligiridis TA, Kintzios S, Yialouris CP (2010) Time-series piecewise evolutionary segmentation based on wavelet transformation and support vector machines. In: Siddiqi AH, Ucan ON, Aslan Z, Oz HH, Zontul M, Erdemir G (Eds) Proceedings of the fifth international symposium on wavelet applications to world problems (IWW-2010), 7–8 June, Istanbul, Turkey, ISBN: 978 650 4303 038
48. Kintzios S, Bem F, Mangana O, Nomikou K, Markoulatos P, Alexandropoulos N, Fasseas C, Arakelyan V, Petrou A-L, Soukouli K, Moschopoulou G, Yialouris C, Simonian A (2004) Study on the mechanism of bioelectric recognition assay: evidence for immobilized cell membrane interactions with viral fragments. Biosens Bioelectron 20:907–916
49. Kintzios S, Goldstein J, Perdikaris A, Moschopoulou G, Marinopoulou I, Mangana O, Nomikou K, Papanastasiou I, Petrou A-L, Arakelyan V, Economou G, Simonian A (2005) The BERA diagnostic system: an all-purpose cell biosensor for the 21th Century. 5th Biodetection Conference, Baltimore, MD, USA
50. Chang C-C, Lin C-J (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2:27:1–27:27