Computers and Electronics in Agriculture 70 (2010) 263–275

Contents lists available at ScienceDirect

Computers and Electronics in Agriculture

journal homepage: www.elsevier.com/locate/compag

Plant virus identification based on neural networks with evolutionary preprocessing

Thomas J. Glezakos[a,*], Georgia Moschopoulou[b,2], Theodore A. Tsiligiridis[a,1], Spiridon Kintzios[b,2], Constantine P. Yialouris[a,1]

[a] Agricultural University of Athens, Department of Science, Informatics Laboratory, Iera Odos 75, 11855 Athens, Greece
[b] Agricultural University of Athens, Agricultural Biotechnology Department, Laboratory of Plant Physiology and Morphology, Iera Odos 75, 11855 Athens, Greece

ARTICLE INFO

Article history:
Received 15 December 2008
Received in revised form
11 September 2009
Accepted 17 September 2009

Keywords:
Plant virus identification
Time-series analysis
Genetic algorithms (GAs)
Artificial Neural Networks (ANNs)
Machine learning
Data mining
Bioelectric Recognition Assay (BERA)
Preprocessing techniques
Plant virus identification
Feature extraction

ABSTRACT

In this work, genetic algorithms and multilayer neural networks are applied to plant virus identification. The initial data set is derived via a well known prototype method, which uses specially designed biosensors to monitor the virus reactions. Several techniques have been introduced for preprocessing the plant virus waves. They include segmentation along the time axis for fast response, nonlinear normalization to emphasize significant information, averaging samples of the plant virus waves to suppress noise effects, reduction in the number of samples to realize a more compact network, etc. Given the features of the acquired virus time-series signals of the problem under study, an evolutionary method is proposed in order to produce meta-data from the original time-series initial information, reduce the dimensionality of the input data space, and to eliminate the noise inherent in the initial raw information. A genetic algorithm is employed so as to smooth out the initial information while, the so produced meta-data sets are used in the training and testing of the applied neural network, producing fitter training data. The method is tested against some of the most commonly used classifiers in machine learning via cross-validation and proved its potential towards assisting virus identification.

1. Introduction

An important problem arising while analyzing large time-series data sets, both in dimension and size, relates to the proper selection of a subset of the original features. Preprocessing the time series to obtain a representative meta-data set not only significantly reduces computational time, but also functions as a smoothing technique to weed out possible non-systematic portions of the initial information, which may, in an extent, inhibit the analytical process. Conventional methods of time-series data preprocessing, such as segmentation along the time axis for fast response, nonlinear normalization to emphasize significant information, averaging samples of the plant virus waves to suppress noise effects, reduction in the number of samples to realize a more compact network, include descriptive statistical methods such as re-sampling techniques or moving average procedures, both of which manipulate the initial information in a fixed width fashion. On the other hand, time-series analysis plays an important role for phytopathology and virology, especially as regards to virus identification, which is made possible due to time-series assessment.

In this manuscript an innovative method is described, aiming to overcome the limitations posed by the fixed width of the analytical tools. The algorithm designed allows for the production of effective meta-data, after having preprocessed the original time-series information in an evolutionary fashion. Thus, it drastically reduces the size of the raw data table to more compact sets of cases and, at the same time, retaining all the crucial information of the initial time series. This is achieved by the development of analytical tools of evolutionary adaptive width, propelled by Genetic Algorithms (GAs) and Neural Networks (NNs). In its present stage the algorithm is applied to the identification of the *Tobacco Rattle Virus* (TRV) and the *Cucumber Green Mottle Mosaic Virus* (CGMMV), using an initial raw time-series data set of large proportions and effectively reducing the length of the input data in each of the above two cases. The developed system aims to overcome the problems posed by the dimensionality and noise of the input data vector on the classification power of NN systems dealing with time series. The classification potential of the proposed system is also

* Corresponding author. Tel.: +30 210 5294181/176/182.
E-mail addresses: t_glezakos@yahoo.com (T.J. Glezakos), geo_mos@aua.gr (G. Moschopoulou), tsili@aua.gr (T.A. Tsiligiridis), skin@aua.gr (S. Kintzios), yialouris@aua.gr (C.P. Yialouris).
[1] Tel.: +30 210 5294181/176/182.
[2] Tel.: +30 210 5294292.

0168-1699/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.compag.2009.09.007

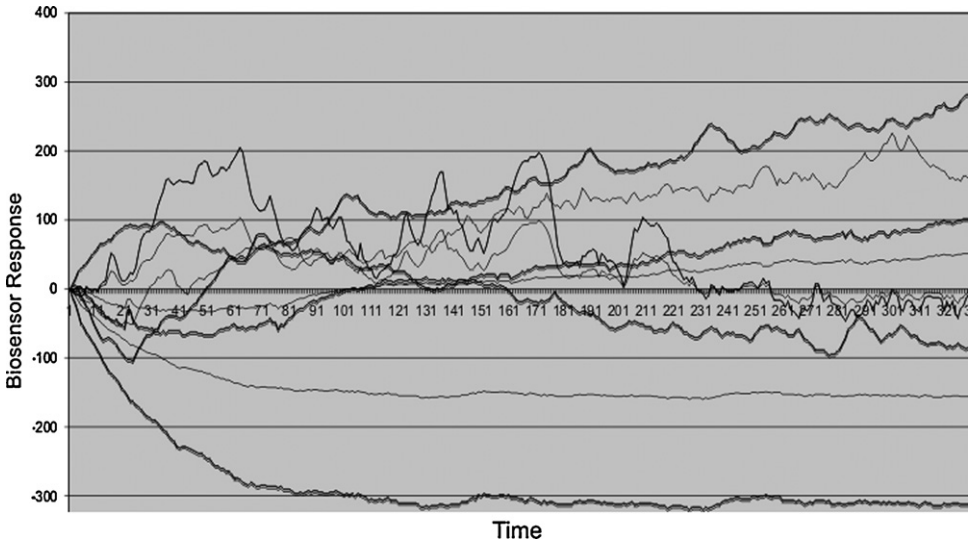

Fig. 1. CGMMV raw time-series data, as obtained by the BERA method.

compared towards several widely used classifiers, trained and tested with the same initial time-series data set. The key feature of the system is the prototyping of the evolutionary procedure under which the meta-data sets are genetically formed, enhancing the identification potential of the NN. Thus, the algorithm initializes by employing several testing techniques to design a multilayered perceptron (MLP) able to use the initial input space as a training pool and produce a rough neural classifier. The variation introduced here and the outmost important contribution of this work is that, instead of training and testing the developed MLP with the whole initial data set, the algorithm alternatively engages an evolutionary method to genetically manipulate the time series.

The manuscript, except from the present introductory part, consists of five more sections. From this point on, the general background regarding the problem is being laid out, interconnecting time-series analysis with the virus identification problems, as well as the potential solutions to them. Also, a brief reference to related work is presented, both regarding time-series feature extraction and virus identification using artificial intelligence tools. The third part summarizes the materials and methods undertaken by the research. Here, a link is attempted among plant virus identification methods, such as the Bio-Electric Recognition Assay (BERA) method and artificial intelligence tools, such as NNs and GAs. In this section, a description of the developed frontend Graphical User Interface (GUI) is also attempted, while at the same time referring to its functional operation. The fourth section of the manuscript gives a detailed description of the inner works of the evolutionary algorithm, developed especially for training meta-data production. Finally, the paper consummates by presenting the results and the conclusions reached in the framework of the research.

2. Background

2.1. Problem essentials

2.1.1. The Cucumber Green Mottle Mosaic and Tobacco Rattle Viruses and their identification

Plant viruses, although not as well understood as their human counterparts, are infectious intracellular parasites, that lack the means of long-term sustenance or reproductive ability, without the supportive existence of the host. Up to now several hundreds of plant viruses have been described and, with time, new ones continue to be reported.

Two of the most well known viruses posing serious economic impact on crops still nowadays are the TRV and the CGMMV, both of which belong to the category of viruses whose infectious material lies in their RNA strands, causing serious productivity pressure on their hosts. The symptoms of the infected plants include notched leaf blade margins, yellow ring-spotting and phenotypic concentric line patterns, mottling, necrotic local lesions, distorted growth, and systemic necrosis. TRV has a host range of over 350 plant species and is transmitted by soil-inhabiting nematodes of the genus *Trichodorus*, mechanically (on infected tools) grafting and seed, while it has not whatsoever been reported to be spread by plant to plant contact. On the other hand, CGMMV pathways include seed transmission and carrying water or soil particles. Their identification and early handling is of the outmost importance for agricultural crops and yield. The general framework of the research presented in this manuscript is to contribute in the plant virus identification problem by combining GA with NNs in order to develop a friendly software tool for their classification.

In this work, we engage the recently introduced BERA method (Kintzios et al., 2001a,b, 2003, 2004, 2005), in order to acquire the initial data set. The products of the method are time series of electrical potential difference, resulting from the virus interaction with properly structured reagents. The two plant virus waves are measured for 331 s, for which, at a sampling rate of 10 Hz, the average value was calculated and recorded for each second. Therefore, 331 average values are obtained for one signal channel corresponding to one virus (signature). Fig. 1 depicts a randomly chosen set of signatures/time series produced by CGMMV, when reacting with the biosensors of the BERA method. The *X*-axis units are time stamps, while *Y*-axis units are measurements produced by the sensor. While reacting to the biosensors, each of the virus waves exhibit patterns of responses (i.e. the amplitude of the virus waves) over specific ranges of concentrations. These responses are thus considered as virus features, a real identification signature, which should be analyzed and classified in order to identify the pathogen in question.

2.1.2. Meta-data motivation

Time-series analysis engages methods which generally formulate two basic analyzing categories: forecasting and identification, both of which encompass processes to understand the driving underlying forces and structures that produced the initially observed data. In most cases the attempt to analyze time-series information is marred by inhibitory factors such as the dimen-

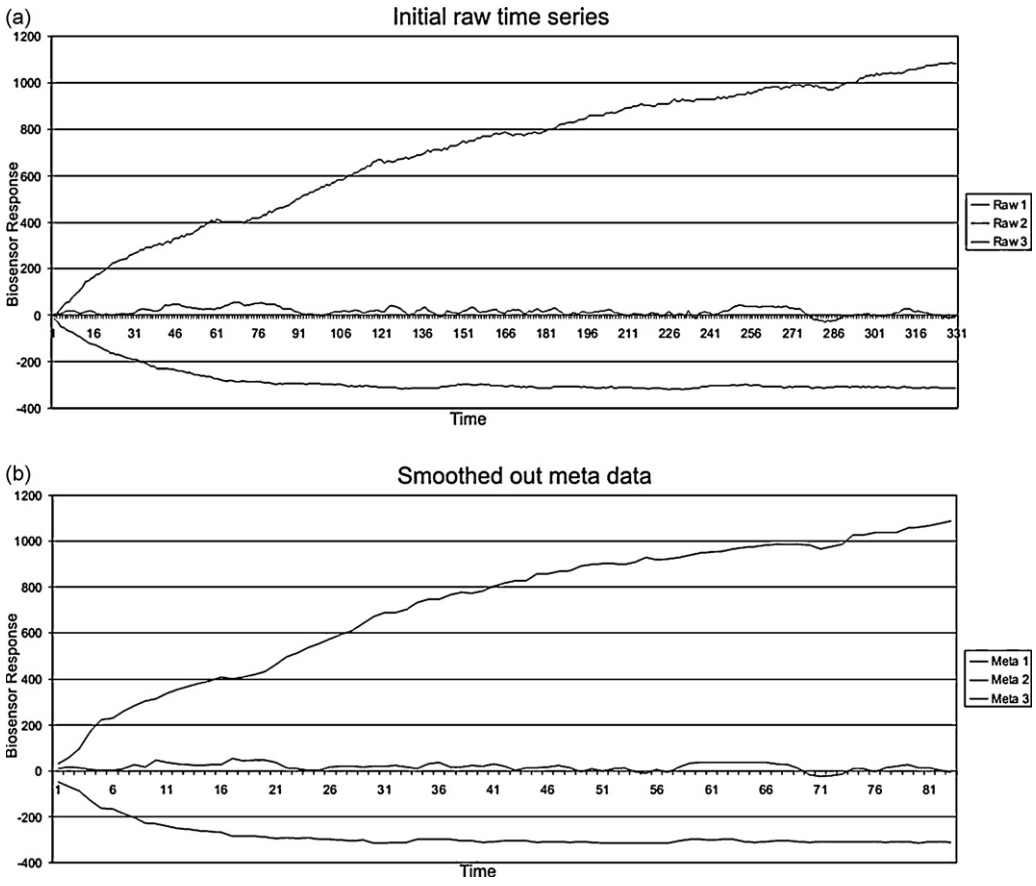

Fig. 2. Raw (a) and genetically smoothed out (b) time-series TRV responses.

sionality of the series, as well as the noise inherent in it. Thus, the data involved are regarded as consisting of two components: a systematic identifiable pattern and random noise in the form of a systematic error, which intervenes and renders the analysis difficult, if not impossible (Box and Jenkins, 1976).

In order to minimize the systematic error it is often essential to preprocess the time series by applying a smoothing technique on the raw time-series information, so as to reduce both the systematic error and the dimensionality of the time series. The proposed system utilizes an evolutionary data segmentation method to achieve this, by producing meta-data which are able to better train the neural classifier. Fig. 2 gives an example of the smoothing procedure results referring to one of the signatures acquired for the TRV. In Fig. 2a the initial raw time-series information is depicted, where each signature comprises of 331 average values, received from the data logger, and the lines seem rather jagged and unsettled. Fig. 2b demonstrates the results of the evolutionary smoothing procedure we followed, depicting the meta-data produced. As we will see, the proposed method achieved to reduce the number of measurements taken from each virus wave to 84 measurements per signature, consequently producing smoother lines.

2.1.3. Problem description: analytical tools' limitations

Exploratory time-series data analysis recently engages a wide range of artificial intelligence tools, such as Artificial Neural Networks (ANNs) and GAs, the proper configuration of which employs in most cases various forms of data preprocessing for the definition of the input vector (Liu et al., 2007). The first and foremost step in this kind of analysis is the smoothing out of the error, using various techniques which almost always engage some kind of local averaging or re-sampling of the time-series sequence, in order to induce mutual cancellation of the non systematic components of individual observations. Applying filtering techniques, most often than not referred to as "smoothing procedures", may remove random variations and reveal trends and cyclic components of the series, provided that their parameterization, which is most crucial in their success (Frank et al., 2001), has been carried out carefully. The most common smoothing techniques include re-sampling, averaging, as well as various exponential smoothing methods, which encompass simple averaging techniques and moving averaging ones, the latter being most effective in the cases that there are trends and signatures inherent in the time-series data.

According to the moving average smoothing technique, each component of the time series is replaced by either the simple or weighted average of n surrounding components. In this case, n is called the width of the smoothing moving window (Box and Jenkins, 1976; Velleman and Hoaglin, 1981). For the re-sampling procedure, the time series is transformed into a new one, consisting of every n-th value of the initial data, in a scope to reduce dimensionality, in which case n is the re-sampling rate. Of course, smoothing medians can replace smoothing means in the first case, having as their main advantage that they are less biased by outliers within the smoothing window, although in the absence of outliers, smoothing medians produce more irregular curves than their averaging counterparts. On the other hand, the width of the moving window, as well as the period of the re-sampling rate, play the most important role not only on the accuracy of the smoothing procedure, but also on the amount of data which is to be stripped off from the initial information. Define it excessively and we may end up with a totally different time series with mass amount of information thrown away and lost, while more conservative definitions may result in an ineffective smoothing procedure and a mostly

noisy result. So far, the definition of the width of such exploratory tools has been defined in a static fashion, that is, the moving average window width does not adapt to the underlying information. In order to overcome this difficulty, the evolutionary data segmentation method we introduce, runs through the time series utilizing a vibrant grouping procedure, which auto-corrects its width according to the data explored and produces meta-data sets which are then used as educational material for a NN classifier to perform the identification. Thus, we attempt to overcome the analytical problems posed by the static definition of the moving average window width, replacing it with an adaptive, dynamically formulated one.

2.2. Related work

In recent literature, numerous are the research projects engaging time-series data preprocessing as a preliminary step for the training of data mining tools, especially neural networks. In 2005, Palmer et al. (2005) developed a machine learning system based on MLP neural networks for tourism time-series forecasting. The research engaged data concerning the Balearic Islands for a period of fourteen successive years from 1986 through 2000. The researchers used logarithmic transformation and differencing to preprocess the time series in order to detect trends, minimize noise, uncover relationships and flatten the variable's distribution, procedure which eventually resulted in a more accurately forecasting MLP. During the same year, Zhang examined the effectiveness of time-series de-seasonalization and de-trending on neural network performance (Zhang and Qi, 2005). Their results clearly indicate that neural networks are not able to model seasonality directly, but they rather need time-series preprocessing, which is critical in order to build an effective neural forecaster. Other types of preprocessing have also been tested. In 2006, Cannas et al. (2006) used discrete wavelet transforms and data partitioning to preprocess time-series data regarding monthly runoff for the Tirso basin, located in Sardinia in Italy. The meta-data were used to train a neural network to forecast runoff one month ahead. Their results show that preprocessing with the discrete wavelet transformation technique produces more effective meta-data than using noisy raw signals. Still in 2006, Simon et al. (2006) conducted research dealing with the usefulness of time-series clustering. Having noted the appearance of papers stating that time-series clustering is meaningless, he contradicts this conclusion by introducing the unfolding preprocessing methodology. The research shows that, provided that adequate preprocessing has taken place, time-series clustering is eventually useful. Recently, Wu et al. (2009) investigated the performance of three time-series preprocessing techniques in the performance of a neural forecaster. Specifically, moving average, singular spectrum analysis and wavelet multi-resolution analysis were engaged to produce meta-data which were compared to unpreprocessed time-series data as regards to their performance as training sets for a neural network estimating daily flows. The results of the study stress the importance of data preprocessing as essential for the production of a fitter meta-data sets. Of the three preprocessing methods, the moving average one gave the best results.

The analysis and feature extraction for time-series data most often than not, resorts to regression models, such as the autoregressive and the moving average methods, which are inhibited by the non-linearity inherent in the input data space. Several cases of ANN systems (Avramidis and Iliadis, 2005; Bodri and Cermak, 2000; Filho and dos Santos, 2006; Harpham and Dawson, 2006; Jain and Kumar, 2007; Kerh and Lee, 2006; Ni and Xue, 2003; Pulido-Calvo and Portela, 2007; Sahoo et al., 2006; Wei et al., 2002; Yadav et al., 2007), Fuzzy Logic (Monfared et al., 2009; Fazel Zarandi et al., 2009; Lai et al., 2009; Kolman and Margaliot, 2009; Fernandez et al., 2008) and Evolutionary Algorithms (Gallant and Aitken, 2003; Hansen et al., 1999), along with Bayesian, Nearest Neighbors and Decision Tree methods (Mora-Lopez et al., 2005; Niu and Yang, 2008; Weng and Shen, 2008; Arroyo and Maté, 2008) are all included into a vast list of methodologies engaged in order to address the aforementioned problems. Also, a lot of research has been dedicated in studying the development and prototyping of NN design or the training and testing via meta-heuristic methods (Abraham, 2004; Elizondo et al., 2007; Niska et al., 2004; Prudencio and Ludermir, 2004; Rossi et al., 2005; Sivagaminathan and Ramakrishnan, 2007). Finally, the multi-classification systems design and prototyping has been contemplated during the recent years, in order to provide models which might be rendered as potential problem solvers. In this latter case, multiple classification techniques and models are combined in individual systems trained to provide solutions to pattern recognition problems. Classifier combination approaches might be categorized mainly over three dimensions in this context: the representational and the architectural methodology, as well as the learning technique (Alpaydin, 1998; Kuncheva, 2000). All these combinatorial methods might potentially provide models able to discern through the input space and offer feasible solutions to problems which, either cannot be solved by single classifier powered systems, or which might be more effectively handled by a multi-classifier one.

As regards to machine learning tools implementation in microbiology, few are the occasions that relate to plant viruses. Morimoto et al. (1997) designed an intelligent control technique engaging NNs and GAs for optimal control of the microbial processes developing during the storage of fruits. Their system linked relative humidity with fruit water loss and increase of the percentage of lesion by fungi in the time unit. In the first step of the process, dynamic responses of water loss and fungal development of fruit as affected by relative humidity were identified in an actual system and then, a model of a single input and two outputs was built using NNs. The inputs of the network received time-series data, while the single output gave a series of values for the objective function they had earlier set. A GA was then engaged to seek out the optimal values which maximized the objective function.

On the contrary, numerous are the studies which engage machine learning tools for zoo-blast virus identification. One year later, Haydon et al. (1998) and his team attempted to identify cirrhosis in patients with chronic hepatitis C virus infection. In their research they trained and validated ANNs with routine host and viral parameters so as to create an automated system trained such as to identify the presence or absence of cirrhosis in the patients. The results showed that the ANN outperformed logistic regression for that matter, reaching at a sensitivity of 92% and a specificity of 98.9%. The predictive values of the positive and negative tests were 95% and 97% respectively, rendering ANNs as potential problem solvers. By the end of the previous century, artificial intelligence had established its potential as a useful tool for microbiological and biochemistry research. Nielsen et al. (1999) used machine learning classifiers in order to predict the pathways that proteins follow en route to their final destinations within a cell, known as sorting signals. The researchers engaged SignalIp, software developed for that purpose and based on NN technology, for the secure prediction of the most well known sorting signal, that of the secretory signal peptide. In the course of the research, the software was enhanced with a hidden Markov chain so as to discriminate between cleaved signal peptides and un-cleaved signal anchors. More recently, Dawson et al. (2006) used an ANN for influenza virus type A surveillance. Specifically, the network was trained so as to recognize influenza fluorescence image patterns, exhibiting high clinical sensitivity and specificity values of 95% and 92% respectively. Also, various NN strands have been engaged in the development of effective computer-augmented pathology tools, helpful in epidemic inhibition or the identification of individuals running a risk of infection. Bayesian networks have been found to be effective in discriminat-

ing between normal from diseased or viral from bacterial tissue samples in mid to late stages of infection (Gilfeather et al., 2007), while identification in earlier stages needs more elaborate time-series analysis tools.

3. Materials and methods

3.1. Development platform

The proposed system is in essence a genetically trained NN with the architecture of a MLP. It was developed using the programming language of Python, enhanced by the freely available open source Fast Artificial Neural Network Library (FANN), which was especially wrapped up for use with the language. The system was built up from scratch and is in an open source state, meaning it is freely and openly available to everyone. Python is a programming language with a relaxed learning curve, while remaining powerful and incorporating efficient high level data structures as well as a simple and effective approach to object oriented programming, allowing for programs to be developed on most of the operating systems available today. The source code was written under Ubuntu Linux, but can be run on a windows–based system just as easily. It is also supported by a vast number of freely available libraries, one of which is FANN, with support for both fully and sparsely connected networks. Cross-platform execution in both fixed and floating point types are supported, while it also includes a framework for easy handling of training data sets. For comparison reasons, the Orange library for Python (Demsar et al., 2004) was utilized so as to construct the classifiers which the proposed system was compared with. It is component-based data mining software including a range of preprocessing, modelling and data exploration techniques which offered the naïve Bayesian, the decision tree and the *k*-Nearest Neighbors (*k*-NN) classifiers which were used as benchmarks in the comparison with the NN. The evolutionary process has been designed in Python from scratch and incorporates a GA in order to produce fitter meta-data out of the initial raw time-series ones. The GA assesses the potential of each designed NN, engaging its fitness function for that matter.

3.2. BERA method

BERA is a recently introduced method using biosensors to identify various chemical and molecular structures. The method essentially assesses the structures' interactions with a group of cell components immobilized in a gel matrix preserving their physiological functions. The structures to be identified are referred to as ligands in the sense that they are usually smaller molecular units – i.e. viruses – which specifically bind to the larger biosensor cells. This procedure ultimately results in the alteration of the reagent physiology and the deliverance of electrical energy. Recent studies (Kintzios et al., 2001a,b, 2003, 2004, 2005) have revealed the usability of the method for cheap and fast identification of human infectious viruses and its potential to replace more time-consuming and costly methods, such as the Reverse Transcription Polymerase Chain Reaction (RT-PCR).

The method is also massively propelled forward due to the fact that the technology behind the biosensor production is advancing by major leaps. After having gone through a number of improving biosensor generations, BERA sensors were radically redesigned to reach the fifth generation of their design development, which incorporates sensors almost ideal for diagnostic applications. In this fifth generation we shall meet sensors of extremely reduced size, consisting of a disposable array of gel beds loaded with reagent cells. Their production is characterized by cost at the lowest levels and a very high rate of reproducibility and speed of manufacturing. Moreover, the fifth sensor generation has achieved to reduce assay time to approximately 12 s. The BERA biosensor method has already been successfully implemented to numerous applications, mainly related to both human and plant virus identification, while its evident potential has been recorded into various bibliographical references (Thach et al., 2003; Boltovets et al., 2004; Skládal et al., 2004; dos Santos Riccardi et al., 2006; Prieto-Simon et al., 2008).

Table 1
Training and testing data sets for the plant identification problem.

Virus	Train set	Test set	Total cases
CGMMV	549	91	640
TRV	530	101	631
Total cases	1079	192	1271

For the purposes of this research, the BERA method was used so as to provide the initial time-series information regarding the two viruses. As it has been pointed out the two viruses' waves are measured for 331 s. That is, for each virus, the data logger of each biosensor was engaged to produce 331 average values, corresponding to measurements taken with a sampling rate of 10 Hz. These average values were used in order to form the signature of the identifiable object. A data set consisting of 1271 cases was thus constructed and used as an instructional scheme for the specially designed evolutionary NN classifier. Table 1 reports on the distribution of the various patterns acquired for the two viruses.

The usual procedure requires the partitioning of the initial data set into the training and testing data sets using a well known dividing scheme, resulting in 1079 and 192 cases, respectively, The BERA method was thus configured so that it utilized special types of biosensors according to the target-specific antibody that was contained on their membrane, specifically for each virus. So, for each example we came up with different biosensor's response data curves. Thus, the overall data set was balanced for this two-class classification problem, consisting in particular of 640 and 631 cases for the CGMM and TR viruses, respectively.

3.3. System parameterization

3.3.1. NN parameterization

ANNs were developed in an attempt to simulate the human cerebral functions (Haykin, 1989). In this context, an ANN is a software device consisting of a number of simple processing elements (neurons) interconnected and operating in parallel. Each neuron is only aware of the signals it receives from other connected neurons and the information it sends from time to time to other processing elements, a procedure enabling the network to learn from examples, through iteration. During the learning process each neuron's synaptic weight vector is repeatedly adjusted in response to stimuli presented as inputs requiring the presence of a known output.

The neural classifier that has been utilized follows the MLP architecture with sparsely connected layers. The code was enhanced with a NN object created from the FANN Python library, which permits for a vast number of parameterization on the *N* that it handles. The neural object developed consists of an input layer of varying number of units, conforming to the meta-data produced during the evolutionary procedure. The most potent of these meta-data are eventually used in the supervised learning of the neural classifier. The variability of the input layer of the neural object was dictated by the GA producing the meta-training data sets, a procedure in which different numbers of input data are produced by each trainer, in an effort to handle the dimensionality of the input space. Thus, it was straightforward to create an elastic input layer, able to adapt to the length of the input vector for each generation. Table 2 summarizes the parameterization used in the neural object of the proposed system.

Table 2
Neural object parameterization.

Parameter	Value	
NN type	Back propagation MLP	
Input layer	Adaptive	
Hidden layer	30 neurons	
Output layer	2 neurons	
Connection rate	Sparse array 80%	
Training algorithm	RPROP	Increase factor 1.5
		Decrease factor 0.5
		Delta min 0.0
		Delta max 0.5
Activation function	Hidden layer	Sigmoid symmetric
	Output layer	Linear piecewise
Activation steepness	0.5	

The decision regarding the use of 30 neurons in the hidden layer of the neural object was assisted by the automatic search procedure offered by the FANN library, known as *cascade training on data*, which determines the optimal number of hidden layers and neurons based on sequential training. Thus, the network starts out its training empty in the hidden layer and then, as training continues, it adds neurons one by one and layer by layer, until an optimal NN structure is reached. The output layer consists of two binary neurons so as to classify the input space to one of the two viruses. Thus, the target for the output has only one non-zero element, such as 10 or 01. The training of the network was dictated by the Rprop training algorithm which outperformed in the evaluating tests both the Incremental Training and the Simple Batch Training algorithms. The Rprop algorithm is a branch of the batch training ones which update the weight table once after the whole epoch has been completed, in contrast to the incremental algorithm which updates the weights immediately after each pattern has been shown to the network.

3.3.2. GA parameterization

GAs on the other hand, are inspired by evolutionary biology, and especially driven by the Darwinian axiom of the "survival of the fittest", incorporating numerous biological procedures such as inheritance, selection, crossover (otherwise referred to as recombination) and mutation. They are mostly implemented as computer simulations which search the input plane for better solutions to a given optimization problem. A GA starts out with an, often random, initial population of encoded representations of candidate solutions. These representations are referred to as chromosomes, genotypes or genome, while the candidate solutions are referred to as phenotypes. The algorithm proceeds by generations each of which is comprised by genotypes of the previous generation, elected basically on their fitness and modified on the grounds of a possible recombination or mutation to form a new population of higher expected fitness. Table 3 reports on the parameterization scheme of the GA.

In its first phase, the system developed during the current work uses the success rate of the neural object to form the fitness function of the GA governing the production of meta-data. As the success rate we considered the ratio of the number of the correctly identified signatures to the number of the total signatures of each meta-data set. The success rate thus calculated is a characteristic of each trainer in each generation and stands for the fitness value of the trainer. The GA seeks to minimize the distance of this fitness value to an issued user defined success rate, set in the initialization stages. In the next and final phase, the neural thus produced is tested using 10-fold cross-validation on testing data which conform to the structure of the winning chromosome of the evolutionary process.

Table 3
GA parameterization.

Parameter	Value
Ideal success rate	95%
Trainer population	12 per generation
Number of generations	500 max
Selection method	Roulette wheel
Mutation rate	5‰
Crossover rate	40%

3.4. Data mining tools used for assessment

The proposed solution was compared as regards to its performance towards three other widely used data mining tools, namely the naïve Bayesian, the decision trees and the *k*-NN classifiers, provided by the Orange library for Python (Demsar et al., 2004).

The naïve Bayesian classifier derives from the Bayes' Theorem. Such classifiers may be trained very efficiently in supervised learning schemes and further be applied successfully to real world problems. A great advantage of these classifiers is that, for most of the problems, but not independent from that, they require relatively smaller training data sets.

The classification tree classifiers, also referred to as decision trees or regression trees, are also predictive models, which engage a mapping of the features affecting the status of a class variable to conclusions about its target value. They essentially are generators of rules referring to the condition of the class they predict, which are clear and can be easily understood and explained. The tree grows up due to a technique which is called 'binary recursive partitioning'. According to this procedure, the input data is iteratively split into partitions of clear meaning. Then each partition is further split into new ones and so on, until a class value is finally met. A tree thus built has leaves for class values and branches for conjunctions of features which lead to these values. Classification trees are models which require supervised learning in order to learn from example and be able to generalize on real world problems. It systematically breaks up the input vector into a number of partitions such that the diversity of the class value is minimal within each partition. This procedure produces homogeneous partitions as regards to the feature class in question. The process is repeated for all the fields of the training set and continued at each next node, until a full tree has been built. In this work we have utilized the pruning for the tree with a factor of 2.

Finally, the *k*-NN classifier was compared against the proposed system. This classifier is ranked among the simplest in apprehension and the hardest in implementation algorithms in machine learning problems. The algorithm is based on the principle of proximity resemblance, i.e. nearest objects are more possible to be alike than farther situated ones. Thus, an object in the input plane is classified according to the majority vote of its *k*-nearest neighbors. The parameter k is the most vital in the implementation of the algorithm, deciding the crucial distance in which proximity resemblance takes effect. Its optimal estimation is very hard to come by and is made feasible only through trial and error techniques or cross-validation. Generally, larger values of k reduce the possibility of noise creeping in classification, but render the boundaries among the values of the class vaguer. In our research, after several tests for k, we have elected $k = 20$.

4. System development

4.1. Meta-data set configuration

Each meta-data set produced by the GA comprises of equal number of cases to the original data set. The only difference is that the number of samples per case (i.e. the virus signature) has been shrunk, according to the evolutionary procedure adopted. Inside

Table 4
Structure of a random trainer chromosome divided in the behavioral mechanism and the activation part.

Behavioral mechanism		Activation part									
1	0	1	0	0	1	1	0	0	1	0	1

each training meta-data set, each case consists of three parts: the first part is produced by mapping the trainer chromosome onto the initial time-series information. The second part, also referred to as structural information, is the reagent utilized by the BERA method in order to produce the virus signature. Lastly, the third part is complemented by the binary output class for the CGMMV and the TR viruses respectively. In the following sub-sections we describe the configuration of the trainer genome, the mapping of each trainer as well as the encoding of the output class.

4.1.1. Trainer configuration

Through the present research we achieve to prototype the production of fit meta-data out of time-series data, by utilizing an innovative GA. The first and subsequent generations of the GA start out by assembling a population of a user defined number of chromosomes, the genome of which consists of randomly chosen binary genes. Each chromosome is used so as to manipulate the input raw time-series data according to their configuration, in order to produce fitter educational meta-data sets used in the supervised learning of the NN. Table 4, depicts the genome of a fictitious chromosome which, for demonstration reasons represents an original time-series data set of 10 values. As shown, the genome of the chromosome is divided in two parts.

The main part consists of a number of randomly chosen binary bits, equal to the time segments of the initial raw time-series data and is called the 'activation part', for it bears as its duty to carry out the instructions given by the behavioral mechanism. The second part of the chromosome, located at the beginning of the genome, is the 'behavioral core mechanism' of the chromosome. It consists of two supplementary random binary bits (0 or 1), the structure of which defines a set of rules deciding the actions taken by the chromosome towards the raw data that it manipulates.

4.1.2. Data manipulation and chromosome mapping

Each training chromosome exists in order to map its genes onto the raw data set, according to the configuration of its core mechanism. For each generation the algorithm produces equal to the user defined number of the chromosomes/trainers meta-data cases. The produced meta-data sets are in turn used for the training of the NN and the best set is ultimately chosen. The mapping of the chromosome onto the raw data set is essentially a descriptive statistical function. Other mapping functions may also be used. Each chromosome behaves as dictated by its core mechanism genes. If these are 00, then the chromosome is of the "Discard-All-Zeros" type and engages a resampling function, with which the raw time series will be stripped off of its values for which the corresponding genes of the trainer is 0. If the core genes are 11, then the chromosome behaves according to a "First-One-Last-Zero Average" clustering function, with which it extracts the average of the initial time-series elements for every group of its own genes defined as starting with the first 1 and ending with the last zero inside the main body of the chromosome. Likewise, if the core genes are 01 or 10, then the chromosome will engage a "First-One-Last-Zero Median" or a "First One Last Zero MinMax" behavior, respectively. These are clustering functions providing the median or the absolute difference of the maximum to the minimum value for the clusters defined as mentioned above. Table 5 gives a brief numerical example of such an evolutionary data mapping for one of the TRV signatures, using a fictional raw time series of ten values and an accordingly formulated trainer/chromosome.

In this example, if the core mechanism genes were 00, then the trainer would function as a Discard-AllZeros resampling procedure, eliminating the corresponding values (32, 17, 12, 1, 30) for which its genes are 0. Thus, we would end up with a meta-data frame such as 44, 8, 8, 18, 48. If the core mechanism genes are 11, then the trainer would function as an averaging procedure, creating segments of data with the first 1 and the last 0 of its genes. In this case, we discern 5 such segments: the first one (1, 0, 0 for the trainer) comprises of the three first values (44, 32, 17) of the raw time series, which are averaged producing the value 31 for the first segment. The next segment is the single 1 (4th gene of the trainer), preserving the value of 8 in the meta-data frame. The following segment comprises of the genes number 5, 6, and 7 of the trainer (1, 0, 0) averaging the corresponding values (8, 12, 1) of the time series to the number 7. The fourth segment comprises of genes number 8 and 9 (1, 0) giving the average of 18 and 30 which is 24, followed by the last single segment of the trainer, the last 1, which preserves the last value of the time series, number 48. In this case we would end up with a meta-data frame of the types 31, 8, 7, 24, 48. In the same sense, the meta-data for the combinations of 01 and 10 of the core mechanism genes are produced by utilizing the median and the max–min distance respectively. By imprinting the structure of the trainers' genome we produce a number of meta-data cases in each generation equal to the number of the trainers' population. These are in turn fed into the neural object and their performance is monitored and acquired, in an attempt to include the most potent chromosomes into the next generation.

After the mapping of the trainer chromosome on the initial raw time-series data, we end up with a series of meta-data sets the number of which equals to the number of the trainers in each generation. Inside each meta-data set, the number of cases is equal to the number of the cases of the initial time-series information. We discern two types of meta-data sets, that is the training meta-data set and the testing one, with similar layouts. The performance of the neural classifier was monitored in the testing phase of the system. The output of the produced testing meta-data sets was known

Table 5
Mapping of trainer chromosomes onto raw initial time-series data, according to their core behavioral genes for the TRV (01).

	Input data										Output
Raw time series	44	32	17	8	8	12	1	18	30	48	01
Chromosome genome	1	0	0	1	1	0	0	1	0	1	
Core genes	Meta-data production (one case)										
00	44			8	8			18		48	01
11	31			8	7			24		48	01
01	32			8	8			24		48	01
10	27			8	11			12		48	01

beforehand, so the success rate of the proposed system was estimated as the ratio of the number of the correct responses from the neural object to the number of the total cases in the testing meta-data set. Note that the training and testing cases do not include the combinations of 11 and 00. Of course the system may output one of four possible combinations, two of which correspond to the two viruses (10 and 01) and only one is correct each time. Given that in the initial raw information acquired by the BERA method, there were no samples not belonging to one of the two viruses (Table 1), the algorithm considers the probable outputs 00 and 11 as failures whatsoever.

4.2. Layout of the algorithm

Fig. 3 depicts the algorithm applied step-by-step.

The developed algorithm initiates by populating its first generation with a number of trainers, the plurality of which is defined by the user. As shown in the previous sections, the trainers are binary vectors, which are considered as chromosomes behaving in certain predefined ways and able to appropriately manipulate the initial raw time series. After the testing phase, each trainer is assigned a fitness value, i.e. the success rate of the NN trained and tested with meta-data produced by the corresponding trainer/chromosome. Thereinafter, the next and subsequent generations are formulated according to a selection policy which elects the fittest members of the previous generation of trainers.

The user also defines the ideal success rate to be approached in the initialization stages of the algorithm. The most potent and powerful trainers are the ones closer to the user defined ideal success rate set in the initialization stage of the algorithm. Thus, after a number of subsequent generations, the algorithm is expected to create a powerful, highly accurate NN through the evolutionary update of its weight vector.

4.3. The survival of the fittest

Each meta-data set comprises of a number of cases equal to the initial raw time-series data, and a number of attributes (i.e. columns) genetically reduced by the mapping of the corresponding trainer. So, after the testing phase in each generation, each trainer is assigned a fitness score, which is derived by the trainer's performance assessed by the success rate of the neural 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 success rate of the NN trained and tested with the meta-data produced by the i-th trainer of the j-th population of the algorithm. Then, the fitness score f_{ij} assigned to the i-th trainer of the j-th population, should be $f_{ij} = 1/|r_{ij} - h|$, where h is the 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 success rate of the neural object is measured during the testing phase, where it is applied on totally unseen cases, i.e. the evaluation meta-data set.

The algorithm stores the meta-data produced, and then updates each trainer with the corresponding fitness score. The policy of selecting the best offspring incorporates the stochastic procedure known as roulette wheel selection. Thus, the fitness score f_{ij} of the i-th trainer ($i = 1, 2, \ldots, V$) of the j-th population ($j = 1, 2, \ldots, M$) has probability $p_{ij} = f_{ij} / \sum_{i=1}^{V} f_{ij}, \forall j = 1, 2, \ldots, M$ of being selected.

It is essential to add at this point that a trainer may be selected more than once to form the next generation of chromosome selection pool. Of course, the frequency of each trainer in this array is proportionate to its potential. After the corresponding fitness score has been assigned to each trainer, the algorithm incorporates a fitness proportionate selection operator, which elects to perpetuate

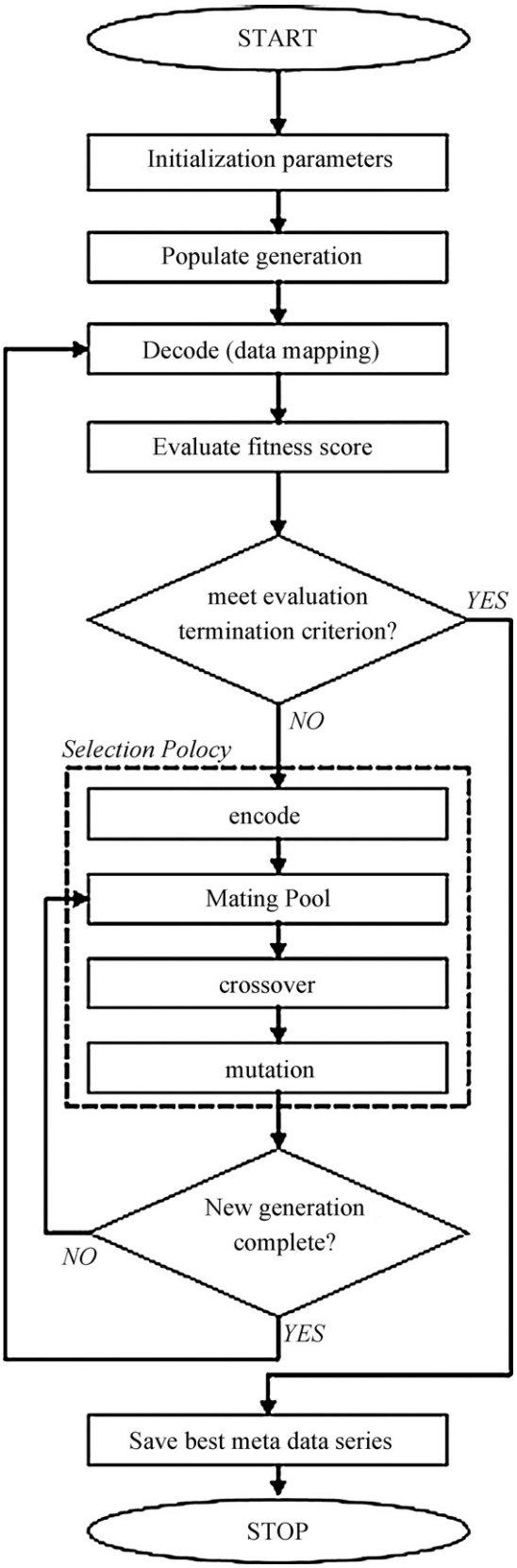

Fig. 3. The course of the evolutionary process.

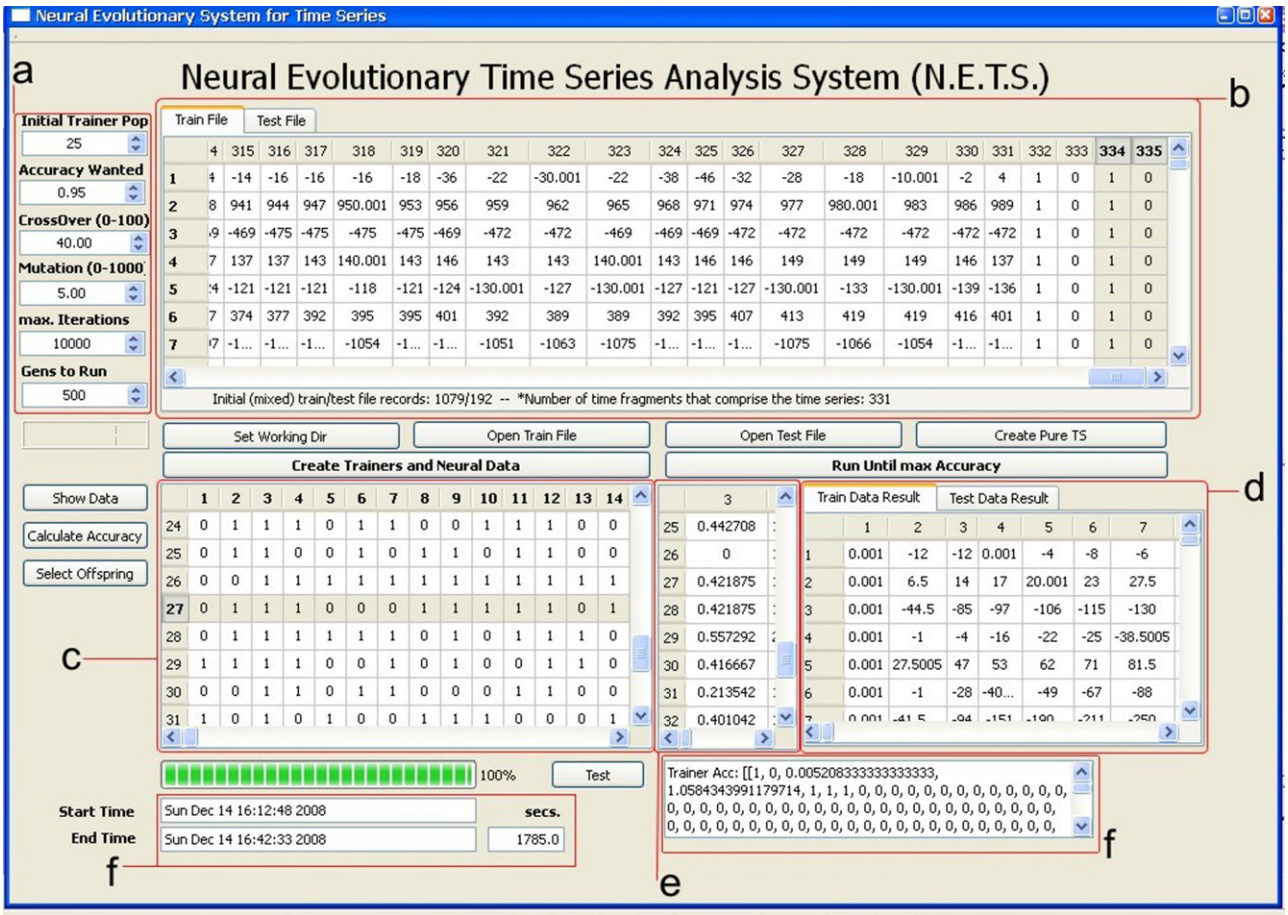

Fig. 4. System frontend interface having loaded plant virus time-series data.

the fittest chromosomes, i.e. the ones with the higher fitness score. In this context, chromosomes with relatively high fitness scores are less likely to be eliminated. On the other hand, less fit chromosomes may not be extinguished from the genetic pools 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. Thus, the algorithm shifts and turns to formulate successive generations, until it succeeds in finding a potent enough meta-data set in order to effectively update the weight vector of the neural object.

4.4. System frontend

The Graphical User Interface (GUI) which was developed is simple but robust. The user is offered the possibility to define right from the start the population of trainer chromosomes, the ideal success rate for the final product, the crossover and mutation probabilities, as well as the maximum iterations and the maximum generations to run, if the ultimate goal has not yet been achieved.

Fig. 4 depicts the developed system frontend. The left-hand side of the panel (Fig. 4a) shows the parameterization of the GA (trainer population, accuracy, mutation/crossover probabilities, etc.) set by the user. In the upper table (Fig. 4b) the train and test data files have been loaded and the time-series portion of the input has been selected. Below and left (Fig. 4c), the 27th trainer has been selected and the interface shows on the right (Fig. 4d) the non time-series meta-data produced for this specific trainer. In the middle (Fig. 4e), the success rate of each trainer in the current generation is being displayed. Information on the duration of the training procedure, as well as the description of the trainers is also provided (Fig. 4f).

The procedure starts out by loading the train and test files containing the initial raw time-series information, along with the value of the attribute class which the system is to be trained for. These training/testing values populate the starting tables at the top of the window, allowing the user to indicate the columns comprising the time series, by selecting them. By pressing the appropriate button, the system will generate the first generation of trainers and map their genes to the time series producing the first generation of meta-data. These will be thrown into the NN and the most efficient record sets will be merged into the next generation.

5. Results and discussion

The work so far presented was essentially motivated by the assumption that the input data acquired by the BERA method might be enhanced by removing any component which mars the discrimination capability of any subsequent machine learning tool. Therefore, there should be a more informative subset of the features of the initial information, producing powerful meta-data for the training of the discriminant NN. To test this assumption, the research proceeded in two major phases:

- First, we used a specially designed GA to smooth out the initial information. This process resulted in the production of a series of evolutionary meta-data sets, which gave better and better results in the course of successive generations.

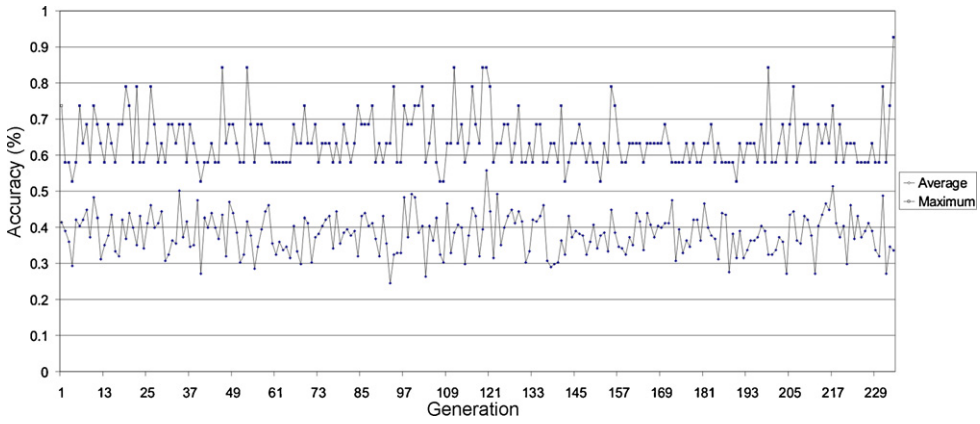

Fig. 5. Performance of the proposed system: maximum and average success rate per generation.

- Once the best trainer was found, a 10-fold cross-validation scheme was employed in order to test the discrimination capabilities of our system towards three of the most popular data mining tools, using the initial raw time-series information, as well as the best meta-data set derived by the evolutionary process.

During the first phase, we seek to create genetically enhanced non-time-series meta-data for the modelling of the responses of the CGMM and the TR viruses. The system, equipped with an NN classification machine as provided by the FANN Python library, initiated the evolutionary production of multiple meta-data sets from the initial time-series information. Assessing each one of these meta-data sets via the discrimination capability of the neural object, the algorithm eventually succeeded in training and testing the latter with the fittest set produced.

The system discrimination potential was derived as regards to the validation subset formed for each meta-data set. The success rate of each trainer was estimated as the ratio of the correctly classified patterns to the overall number of patterns in the validation data set formed by the same trainer. Inside each generation, the best trainer performance as well as the average performance of all trainers was estimated, in order to monitor the course of the evolutionary process, which is depicted in Fig. 5.

In the course of our testing experiments the NN achieved a success rate of 93% in the 234th generation. Table 6 reports on the five best accuracies and the generations in which they were accomplished.

The second step in our research design was to compare the discrimination potential of the derived best meta-data set towards the initial raw time-series information. We also tested the proposed solution towards a range of widely used classifiers, chosen among the most common in machine learning, so as to assess its robustness. For each classification method, a single metric was estimated, namely the success rate of the classifier, which was calculated as the effective generalization of each one, expressed as the ratio of correctly classified input patterns to the total number of presented patterns during the testing phase.

Table 6
Trainer chromosome accuracies for the plant virus identification problem.

Generation	Trainer number	Success rate
234	3	0.93%
46	12	0.84%
19	9	0.79%
1	12	0.74%
8	3	0.64%

For this purpose we utilized a 10-fold cross-validation scheme, in which the two data sets to be compared were partitioned into ten complementary subsets each, while the neural classifier started untrained. The training was then performed on one subset, which included nine partitions, while the testing of the system was done with the remaining tenth partition. The procedure went on until all partitions had served as a testing subset and the result was averaged (Table 7).

In this last evaluating procedure we also compared the performance of the proposed solution towards the potential of three other popular classification systems. The comparison shows (Table 7) that the neural classifier trained and tested with the derived best meta-data set outperforms not only itself when the initial information was used, but also all other classifiers, regardless of the data set used for training.

The results of the testing showed that:

- During the course of the evolutionary process, a multitude of meta-data sets were produced. In fact, until the fittest trainer

Table 7
Cross-validation results for the proposed method and three widely used classifiers.

	Input: original time-series data				Input: best trainer meta-data			
	Neural	Bayes	Tree	*k*-NN	Neural	Bayes	Tree	*k*-NN
Fold-1	0.967	0.688	0.867	0.844	0.956	0.789	0.915	0.901
Fold-2	0.236	0.717	0.866	0.858	0.778	0.704	0.845	0.859
Fold-3	0.847	0.787	0.874	0.913	0.847	0.718	0.901	0.831
Fold-4	0.444	0.756	0.921	0.921	0.972	0.704	0.930	0.845
Fold-5	1.000	0.669	0.882	0.906	0.935	0.775	0.930	0.873
Fold-6	0.097	0.748	0.898	0.890	0.913	0.718	0.901	0.901
Fold-7	0.806	0.622	0.858	0.866	0.917	0.746	0.887	0.930
Fold-8	0.792	0.717	0.898	0.898	0.869	0.746	0.944	0.915
Fold-9	0.083	0.677	0.913	0.882	0.960	0.803	0.871	0.929
Fold-10	0.972	0.661	0.874	0.858	0.847	0.831	0.786	0.871
Average	0.624	0.704	0.885	0.884	0.899	0.754	0.891	0.886

was found, 5850 such sets had been produced comprising of 1271 cases of varying length each, according to the instructions of the corresponding trainer used in their formation. The neural object was trained with each one of these meta-data and the success rate of the test phase was assigned as a fitness score to the corresponding trainer. These facts, combined with the interpreted nature of Python, had as a result that the time of training through to the optimal solution was quite long on a 2.4 GHz Pentium PC with 1 Gb of RAM.

- The GA leads to the construction of a fitter meta-data set. During the evolutionary process, the main focus lay in effectively reducing the features of the initial time-series data. Thus, it concluded in providing a fitter trainer which formed a meta-data set of 84 time stamps in the 234th generation, providing a testing success rate as high as 93% (Fig. 5).
- There is a 'key generation' among the 234 generations that the GA materialized in order to conclude on the fittest trainer. As shown in Fig. 5, the decisive generation for the performance of the system and the one which fills its genetic pool with fit genetic material is generation 120, where an average success rate of 0.56 is noted, along with a maximum rate of 0.84. Although such high maximum rates also appear in earlier generations (46, 53, 111 and 119), they are not accompanied by equally high average ones. Thus, generation 120, despite the fact that it is followed by a series of rather poor performing generations, sets the landmark for the success of the evolutionary training.
- The initial time-series information as derived by the BERA method actually contains noise, the elimination of which produces more powerful training data sets. Table 7 reports on results derived by cross-validation of the proposed method and three widely used classifiers (naïve Bayes, decision tree and *k*-NN), comparing the classification potential of the derived best meta-data set towards that of the initial time series. The results show the increase of performance to all classifiers using meta-data formed by the best trainer. Although the three benchmarking classifiers behave somewhat indifferently towards the use of the two data sets, they nevertheless show a slight increase in their performance. The case is totally different with our neural classifier, which exhibits a 45% increase of its discriminant capability when trained with the best meta-data (success rate of 0.899), compared to the initial raw time-series training set (success rate of 0.624).
- The neural network exhibits the most powerful discriminant capability when combined with the products of the best trainer. In other words, the leap in performance has a great impact in the ranking of the proposed solution among the four contestant classifiers. Table 7 reports that the neural classifier is the worst of the four in discriminating the two viruses, when using the initial raw time-series information. This situation is totally reversed when the best meta-data training set is employed. The outcome of the comparison yields that by applying an evolutionary segmentation method on the time-series raw data, in order to produce genetically enhanced meta-data, we are able to produce a more robust classifier with better generalization potential.

6. Conclusions

Effective virus epidemics confrontation, prior to propose and undertake the proper treatment measures, mainly lies on the determination whether the pathogenic organism is a virus of a totally new species or belongs to an already known and described family, procedure known as virus identification. In this context, precise virus identification plays a crucial role in decision making and effective amass of healthy crops and products. In this work we used the recently introduced BERA method, in order to acquire the initial information regarding the identification of two viruses of high economical importance, the *Tobacco Rattle Virus* and the *Cucumber Green Mottle Mosaic Virus*, both of which pose serious productivity problems on their hosts. BERA uses special biosensors containing certain reagents suspended in a gel matrix which, while interacting with the virus particles, produce electrical signals measured as a voltage. This interaction between the biosensor and the virus particles lasts for a certain period of time, thus the voltage series produced are essentially time-series data which are considered as a signature, each of which is a characteristic of the virus and thus should be analyzed in order to determine the pathogen in question.

The proposed solution combines the basics of time-series analysis with NNs and GAs in order to design, develop and test an automated evolutionary technique to preprocess these time series in the training phase of the neural object, so as to produce genetically enhanced meta-data which effectively control the dimensionality of the input space. In this way we are able to find the scheme inside the time-series length which produces the meta-data set with the most promising structure. The outmost benefit offered by the present research is a framework for the development of an evolving system capable of correctly classifying virus signatures. The proposed solution, alongside with its usefulness as a powerful assistant at the hands of virus identification experts, also keeps the effort and expenses required for its function at acceptable levels, as its only demands lie on the acquisition of the initial information, as well as the discovery of the best meta-data set. The former is a well-established methodology, while the latter poses strains only in computational power. The vast number of the produced meta-data sets results in the enhancement of variability inside the genetic pool and thus increases the probability of discovering a fitter combination of genes, towards the uncovering of the fittest structure of the trainer. Once the evolutionary procedure concludes in finding the best meta-data set, this will always consist of a subset of the features of the initial information. Thus, the proposed system achieves an effective reduction in the dimensionality and the production of a more lightweight input vector with obvious benefits, as regards not only to the training time needed, but also to the accuracy of the developed system.

Of course, once trained, the proposed system exhibits the same classification time demands as the rest of the classifiers and, of course, is very competitive to the time an expert needs to make a decision by evaluating a signature curve. Also, the proposed system yet remains to be comprehensively validated by human experts. Such a procedure is undergone at this time and, although we are not yet able to provide a concise answer, however some of the preliminary results obtained so far are encouraging. Another difficulty we encountered was that the developed GA seemed particularly sensitive to slight changes in its parameterization. Its fine tuning was very difficult as it demanded a great amount of time and trial and error interventions.

Yet in the first stages of system development, the floating point representation of the chromosome genes was rejected and the binary representation was adopted instead, for the problems posed by the former were impossible to come by at the present stage. However, it appears as a good prospect for future work, due to its many virtues, such as greater variability assembled inside the genetic pool or greater resistance against trapping into local minima. The system, as presented here, may be successfully applied to classification problems and, with slight changes in the core engine, to regression problems as well. What is yet unknown is the amount of data that it can handle in the search for the best trainer to each category of problems. The plans of the research team for the near future are to conduct a research project in order to find the upper limits of the system as regards to the maximum length of the ini-

tial information which it can handle, not only in terms of number of cases, but also as regards to each case length, i.e. time-series values, as well. It would also be interesting to research the possibility of widening the scope of the algorithm by interweaving other data mining tools as fitness score providers, besides the neural network. It is clear at this point that such an alteration may require a heavy re-engineering of the source code to control the computational demands of such a product. Finally, a professionally designed GUI might contribute in the wider acceptance of the application by the average user.

Acknowledgements

We would like to thank the authors of Python (http://www.python.org), Orange library (http://www.ailab.si/orange/) and FANN library (http://www.leenissen.dk/fann/) for their efforts in providing such elaborate software as open source.

References

Abraham, A., 2004. Meta learning evolutionary artificial neural networks. Neurocomputing 56, 1–38.
Alpaydin, E., 1998. Techniques for combining multiple learners. In: Proceedings of Engineering of Intelligent Systems, vol. 2. ICSC Press, pp. 6–12.
Arroyo, J., Maté, C., 2008. Forecasting histogram time series with *k*-nearest neighbours methods. International Journal of Forecasting, doi:10.1016/j.ijforecast.2008.07.003.
Avramidis, S., Iliadis, L., 2005. Wood–water isotherm prediction with artificial neural networks: a preliminary study. In: Holzforschung. Walter De Gruyter & Co., Berlin, New York, ISSN: 0018-3830 59 (3), pp. 336–341.
Bodri, L., Cermak, V., 2000. Prediction of extreme precipitation using a neural network: application to summer flood occurrence in Moravia. Advances in Engineering Software 31, 311–321.
Boltovets, P.M., Snopok, B.A., Boyko, V.R., Shevchenko, T.P., Dyachenko, N.S., Shirshov, Yu.M., 2004. Detection of plant viruses using a surface plasmon resonance via complexing with specific antibodies. Journal of Virological Methods 121, 101–106.
Box, G., Jenkins, G., 1976. Time Series Analysis: Forecasting and Control, revised ed. Holden-Day, Oakland, CA.
Cannas, B., Fanni, A., See, L., Sias, G., 2006. Data preprocessing for river flow forecasting using neural networks: wavelet transforms and data partitioning. Physics and Chemistry of the Earth 31, 1164–1171.
Dawson, E.D., Moore, C.L., Smagala, J.A., Dankbar, D.M., Mehlmann, M., Townsend, M.B., Smith, C.B., Cox, N.J., Kuchta, R.D., Rowlen, K.L., 2006. MChip: a tool for influenza surveillance. Analytical Chemistry 78, 7610–7615.
Demsar, J., Zupan, B., Leban, G., 2004. Orange: From Experimental Machine Learning to Interactive Data Mining, White Paper. Faculty of Computer and Information Science, University of Ljubljana (http://www.ailab.si/orange).
dos Santos Riccardi, C., Dahmouche, K., Santilli, C.V., da Costa, P.I., Yamanaka, H., 2006. Immobilization of streptavidin in sol–gel films: application on the diagnosis of hepatitis C virus. Talanta 70, 637–643.
Elizondo, D.A., Birkenhead, R., Gongora, M., Taillard, E., Luyima, P., 2007. Analysis and test of efficient methods for building recursive deterministic perceptron neural networks. Neural Networks 20, 1095–1108.
Fazel Zarandi, M.H., Rezaee, B., Turksen, I.B., Neshat, E., 2009. A type-2 fuzzy rule-based expert system model for stock price analysis. Expert Systems with Applications 36, 139–154.
Fernandez, F.J., Seco, A., Ferrer, J., Rodrigo, M.A., 2008. Use of neurofuzzy networks to improve wastewater flow-rate forecasting. Environmental Modelling & Software, doi:10.1016/j.envsoft.2008.10.010.
Filho, A.J.P., dos Santos, C.C., 2006. Modeling a densely urbanized watershed with an artificial neural network, weather radar and telemetric data. Journal of Hydrology 317, 31–48.
Frank, R.J, Davey, N., Hunt, S.P., 2001. Time Series Prediction and Neural Networks, Journal of Intelligent and Robotic Systems, vol. 31, Issue 1–3. Kluwer Academic Publishers, pp. 91–103.
Gallant, P.J., Aitken, G.J.M., 2003. Genetic algorithm design of complexity-controlled time-series predictors. In: 0-7803-8178-5/03, IEEE XIII Workshop on Neural Networks for Signal Processing.
Gilfeather, F., Hamine, V., Helman, P., Hutt, J., Loring, T., Lyons, C.R., Veroff, R., 2007. Learning and modeling biosignatures from tissue images. Computers in Biology and Medicine 37, 1539–1552.
Hansen, J.V., McDonald, J.B., Nelson, R.D., 1999. Time series prediction with genetic algorithm designed neural networks: an empirical comparison with modern statistical models. Computational Intelligence 15 (3).
Harpham, C., Dawson, C.W., 2006. The effect of different basis functions on a radial basis function network for time series prediction: a comparative study. Neurocomputing 69, 2161–2170.
Haydon, G.H., Jalan, R., Ala-Korpela, M., Hiltunen, Y., Hanley, J., Jarvis, L.M., Ludlum, C.A., Hayes, P.C., 1998. Prediction of cirrhosis in patients with chronic hepatitis C infection by artificial neural network analysis of virus and clinical factors. Journal of Viral Hepatitis 5, 255–264.
Haykin, S., 1989. Neural Networks, A Comprehensive Foundation. Prentice Hall International, New Jersey.
Jain, A., Kumar, A.M., 2007. Hybrid neural network models for hydrologic time series forecasting. Applied Soft Computing 7, 585–592.
Kerh, T., Lee, C.S., 2006. Neural networks forecasting of flood discharge at an unmeasured station using river upstream information. Advances in Engineering Software 37, 533–543.
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. Biosensors & Bioelectronics 20, 907–916.
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 21st Century. In: 5th Biodetection Conference, Baltimore, MD, USA.
Kintzios, S., Makrygianni, E.F., Pistola, E., Panagiotopoulos, P., Economou, G., 2003. Effect of amino acids and amino acid analogues on the in vitro expression of glyphosate tolerance in johnsongrass (*Sorghum halepense L.* pers.). Journal of Food, Agriculture and Environment 3, 180–184.
Kintzios, S., Pistola, E., Konstas, J., Bem, F., Matakiadis, T., Alexandropoulos, N., Biselis, I., Levin, R., 2001a. Application of the bioelectric recognition assay (BERA) for the detection of human and plant viruses: definition of operational parameters. Biosensors and Bioelectronics 16, 467–480.
Kintzios, S., Pistola, E., Panagiotopoulos, P., Bomsel, M., Alexandropoulos, N., Bem, F., Biselis, I., Levin, R., 2001b. Bioelectric recognition assay (BERA). Biosensors and Bioelectronics 16, 325–336.
Kolman, E., Margaliot, M., 2009. Extracting symbolic knowledge from recurrent neural networks—a fuzzy logic approach. Fuzzy Sets and Systems 160, 145–161.
Kuncheva, L., 2000. Combining classifiers by clustering, selection and decision templates. Technical report. University of Wales, UK.
Lai, R.K., Fan, C.Y., Huang, W.H., Chang, P.C., 2009. Evolving and clustering fuzzy decision tree for financial time series data forecasting. Expert Systems with Applications 36, 3761–3773.
Liu, F., Ng, G.S., Quek, C., 2007. RLDDE: a novel reinforcement learning-based dimension and delay estimator for neural networks in time series prediction. Neurocomputing 70, 1331–1341.
Monfared, M., Rastegar, H., Kojabadi, H.M., 2009. A new strategy for wind speed forecasting using artificial intelligent methods. Renewable Energy 34, 845–848.
Mora-Lopez, L., Mora, J., Morales-Bueno, R., Sidrach-de-Cardona, M., 2005. Modeling time series of climatic parameters with probabilistic finite automata. Environmental Modelling & Software 20, 753–760.
Morimoto, T., de Baerdemaeker, J., Hashimoto, Y., 1997. An intelligent approach for optimal control of fruit-storage process using neural networks and genetic algorithms. Computers and Electronics in Agriculture 18, 205–224.
Ni, J.R., Xue, A., 2003. Application of artificial neural network to the rapid feed back of potential ecological risk in flood diversion zone. Engineering Applications of Artificial Intelligence 16, 105–119.
Nielsen, H., Brunak, S., von Heijne, G., 1999. Machine learning approaches for the prediction of signal peptides and other protein sorting signals. Protein Engineering 12 (1), 3–9.
Niska, H., Hiltunen, T., Karppinen, A., Ruuskanen, J., Kolehmainen, M., 2004. Evolving the neural network model for forecasting air pollution time series. Engineering Applications of Artificial Intelligence 17, 159–167.
Niu, G., Yang, B.-S., 2008. Dempster–Shafer regression for multi-step-ahead time-series prediction towards data-driven machinery prognosis. Mechanical Systems and Signal Processing, doi:10.1016/j.ymssp.2008.08.004.
Palmer, A., Montano, J.-J., Sese, A., 2005. Designing an artificial neural network for forecasting tourism time series. Tourism Management 27, 781–790.
Prieto-Simon, B., Cortina, M., Campas, M., Calas-Blanchard, C., 2008. Electrochemical biosensors as a tool for antioxidant capacity assessment. Sensors and Actuators B Chemical 129 (1), 459–466.
Prudencio, R.B.C., Ludermir, T.B., 2004. Meta-learning approaches to selecting time series models. Neurocomputing 61, 121–137.
Pulido-Calvo, L., Portela, M.M., 2007. Application of neural approaches to one-step daily flow forecasting in Portuguese watersheds. Journal of Hydrology 332, 1–15.
Rossi, F., Delannay, N., Conan-Guez, B., Verleysen, M., 2005. Representation of functional data in neural networks. Neurocomputing 64, 183–210.
Sahoo, G.B., Ray, C., de Carlo, E.H., 2006. Use of neural network to predict flash flood and attendant water qualities of a mountainous stream on Oahu, Hawaii. Journal of Hydrology 327, 525–538.
Simon, G., Lee, J.A., Verleysen, M., 2006. Unfolding preprocessing for meaningful time series clustering. Neural Networks 19, 877–888.
Sivagaminathan, R.K., Ramakrishnan, S., 2007. A hybrid approach for feature subset selection using neural networks and ant colony optimization. Expert Systems with Applications 33, 49–60.
Skládal, P., dos Santos Riccardi, C., Yamanaka, H., da Costa, P.I., 2004. Piezoelectric biosensors for real-time monitoring of hybridization and detection of hepatitis C virus. Journal of Virological Methods 117, 145–151.
Thach, D.C., Shaffer, K.M., Ma, W., Stenger, D.A., 2003. Assessing the feasibility of using neural precursor cells and peripheral blood mononuclear cells for detection of bioactive Sindbis virus. Biosensors and Bioelectronics 18, 1065–1072.

Velleman, P.W., Hoaglin, D.C., 1981. Applications, Basics, and Computing of Exploratory Data Analysis. Duxbury Press, Boston.
Wei, Y., Xu, W., Fan, Y., Tasi, H.T., 2002. Artificial neural network based predictive method for flood disaster. Computers and Industrial Engineering 42, 383–390.
Wu, C.L., Chau, K.W., Li, Y.S., 2009. Methods to improve neural network performance in daily flows prediction. Journal of Hydrology 372, 80–93. Weng, X., Shen, J., 2008. Detecting outlier samples in multivariate time series dataset. Knowledge-Based Systems 21, 807–812.
Yadav, R.N., Kalra, P.K., John, J., 2007. Time series prediction with single multiplicative neuron model. Applied Soft Computing 7, 1157–1163.
Zhang, G.P., Qi, M., 2005. Neural network forecasting for seasonal and trend time series. European Journal of Operational Research 160, 501–514.