Raport Badawczy

Research Report

RB/78/2011

A hybrid approach to dimension reduction in classification

M. Krawczak, G. Szkatuła

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences

Control and Cybernetics

vol. **40** (2011) No. 2

A hybrid approach to dimension reduction in classification[*][†]

by

Maciej Krawczak and Grażyna Szkatuła

Systems Research Institute, Polish Academy of Sciences
ul. Newelska 6, 02-776 Warsaw, Poland
e-mail: {krawczak, szkatulg}@ibspan.waw.pl

Abstract: In this paper we introduce a hybrid approach to data series classification. The approach is based on the concept of aggregated upper and lower envelopes, and the principal components here called 'essential attributes', generated by multilayer neural networks. The essential attributes are represented by outputs of hidden layer neurons. Next, the real valued essential attributes are nominalized and symbolic data series representation is obtained. The symbolic representation is used to generate decision rules in the *IF... THEN...* form for data series classification. The approach reduces the dimension of data series. The efficiency of the approach was verified by considering numerical examples.

Keywords: data series, dimension reduction, envelopes, essential attributes, heteroassociation, machine learning from examples, decision rules, classification.

1. Introduction

In many areas, such as medicine, finance, industry, climate etc., data series arise. The generated data must be registered, stored, transmitted and then analysed. The majority of data series research focuses on the following problems: indexing (e.g. Keogh, Chakrabarti, Pazzani, 2001), clustering (e.g. Keogh, Pazzani, 2001; Wu, Chang, 2004; Krawczak, Szkatuła, 2010c), classification (e.g. Nanopoulos, Alcock, Manolopoulos, 2001; Krawczak, Szkatuła, 2010a,b), summarization (e.g. Lin et al., 2001), and anomaly detection (Shahabi, Tian, Zhao, 2000). Due to a huge amount of data, different kinds of data series representations were developed. In the literature one can find specialized algorithms dealing with such problems, e.g. including decision trees (Rodriguez, Alonso, 2004), neural

[*]Submitted: July 2010; Accepted: May 2011.

[†]The research was partially supported by the Ministry of Science and Higher Education (Poland) under Grant Nr N N519 384936.

networks (Nanopoulos, Alcock, Manolopoulos, 2001), Bayesian classifiers (Wu, Chang, 2004), etc. Some representations are general enough to be used in the mentioned problems, and some are rather specialized, meant for prescribed applications. It is worth mentioning that there is an increasing interest in data series mining (Xi et al., 2006).

It should also be noticed that the machine learning methods from examples creating decision rules could be applied for solving data series classification problems (Krawczak, Szkatuła, 2008).

Data series are often very large. Most of papers on data series mining deal with the problem of reducing dimensionality. Several data series representations were introduced, including the discrete Fourier transform (Faloutsos, Ranganathan, Manolopulos, 1994), the discrete wavelet transform (Chan, Fu, 1999), the piecewise constant models (Keogh, Chakrabarti, Pazzani, 2001), the singular value decomposition models (Keogh, Chakrabarti, Pazzani, 2001), the symbolic aggregate approximation (Lin, Keogh, Lonardi, 2007), and the upper and lower envelopes (Krawczak, Szkatuła, 2010a).

In the present paper our aim was to develop the idea of upper and lower envelopes of data series for classification, first introduced by Krawczak and Szkatuła (2010a, b), and then we propose a novel hybrid approach for reducing dimensionality of the original data series for classification.

The upper and/or lower envelopes give a new data series representation, and the aggregated envelopes, besides the new representation, give a significant dimension reduction of data series (Section 2).

The developed hybrid approach involves, in addition to the envelopes, the following artificial intelligence techniques.

On the basis of aggregated envelopes (upper or lower, or both) generation of essential attributes for data series is introduced. The essential attributes constitute another new representation, with additional dimension reduction of the data series. In order to generate the essential attributes a heteroassciative memory based on artificial neural networks is applied (Section 3). A three-layer feedforward neural network with one hidden layer allows to compress data (e.g., Krawczak, 2003a, b), and the outputs of the hidden layer neurons constitute just the essential attributes.

In order to simplify long data series representation we changed the real values of the essential attributes into nominal values from a short set of possible nominal values. In this way we obtained a simple symbolic representation of the original data series.

Another artificial intelligence technique applied in this paper is generation of decision rules of the following form: 'IF *some conditions are satisfied* THEN *the data series belongs to a proper class*' (Section 4). Generation of decision rules is based on machine learning and the rules constitute a simple model of data series classification.

The proposed hybrid approach of data series class descriptions in the form of rules seems to be more legible than others, and, what is most important, the approach preserves the character of data series sufficient for proper classification. In Section 5 the basic elements of the proposed hybrid system are presented. Section 6 gives examples of practical application of the proposed methodology.

2. The concept of upper and lower envelopes

Suppose we have the following data series

$$\{x_k(n)\}_{k=1}^{k=K} = [x_1(n), x_2(n), \ldots, x_K(n)] \tag{1}$$

for $n = 1, 2, \ldots, N$.

Before starting the dimension reduction of the series (1), each series is normalized to have mean equal zero and standard deviation equal one, because it is obvious that it is meaningless to consider data series with different offsets and amplitudes.

Following the idea borrowed from the signal processing theory we introduced piecewise constant upper and lower envelopes of data series. We applied the piecewise constant functions, also called step functions, with equal length of steps. The length of steps is denoted m-step, meaning that m sample rates of the data series constitute one step.

The so called the m-step upper and m-step lower envelopes, $m << K$, constitute a kind of approximations of (1) in the following way: the m-step upper approximation (envelope) of data series (1), precisely denoted by $\{x_k^2(n)\}_{k=1}^{k=\lfloor \frac{K}{m} \rfloor m}$, has the following forms:

$$\begin{aligned}
&x_{1}^2(n) = \max\{x_{1}(n), x_{2}(n), \ldots, x_{m}(n)\} \\
&x_{2}^2(n) = \max\{x_{1}(n), x_{2}(n), \ldots, x_{m}(n)\} \\
&\ldots \\
&x_{m}^2(n) = \max\{x_{1}(n), x_{2}(n), \ldots, x_{m}(n)\} \\
&x_{m+1}^2(n) = \max\{x_{m+1}(n), x_{m+2}(n), \ldots, x_{2m}(n)\} \\
&x_{m+2}^2(n) = \max\{x_{m+1}(n), x_{m+2}(n), \ldots, x_{2m}(n)\} \\
&\ldots \\
&x_{2m}^2(n) = \max\{x_{m+1}(n), x_{m+2}(n), \ldots, x_{2m}(n)\} \\
&\ldots \\
&x^2_{\lfloor \frac{K}{m} \rfloor m-m+1}(n) = \max\{x_{\lfloor \frac{K}{m} \rfloor m-m+1}(n), x_{\lfloor \frac{K}{m} \rfloor m-m+2}(n), \ldots, x_{\lfloor \frac{K}{m} \rfloor m}(n)\} \\
&x^2_{\lfloor \frac{K}{m} \rfloor m-m}(n) = \max\{x_{\lfloor \frac{K}{m} \rfloor m-m+1}(n), x_{\lfloor \frac{K}{m} \rfloor m-m+2}(n), \ldots, x_{\lfloor \frac{K}{m} \rfloor m}(n)\} \\
&\ldots \\
&x^2_{\lfloor \frac{K}{m} \rfloor m}(n) = \max\{x_{\lfloor \frac{K}{m} \rfloor m-m+1}(n), x_{\lfloor \frac{K}{m} \rfloor m-m+2}(n), \ldots, x_{\lfloor \frac{K}{m} \rfloor m}(n)\}.
\end{aligned} \tag{2}$$

Similarly, the m-step lower approximations denoted by $\{x_k^3(n)\}_{k=1}^{k=\lfloor \frac{K}{m} \rfloor m}$ have the following forms:

$$
\begin{aligned}
&x^3_{1}(n) = \min\{x_{1}(n), x_{2}(n), \ldots, x_{m}(n)\}\\
&x^3_{2}(n) = \min\{x_{1}(n), x_{2}(n), \ldots, x_{m}(n)\}\\
&\ldots\\
&x^3_{m}(n) = \min\{x_{1}(n), x_{2}(n), \ldots, x_{m}(n)\}\\
&x^3_{m+1}(n) = \min\{x_{m+1}(n), x_{m+2}(n), \ldots, x_{2m}(n)\}\\
&x^3_{m+2}(n) = \min\{x_{m+1}(n), x_{m+2}(n), \ldots, x_{2m}(n)\}\\
&\ldots\\
&x^3_{2m}(n) = \min\{x_{m+1}(n), x_{m+2}(n), \ldots, x_{2m}(n)\} \qquad (3)\\
&\ldots\\
&x^3_{\lfloor \frac{K}{m} \rfloor m-m+1}(n) = \min\{x_{\lfloor \frac{K}{m} \rfloor m-m+1}(n), x_{\lfloor \frac{K}{m} \rfloor m-m+2}(n), \ldots, x_{\lfloor \frac{K}{m} \rfloor m}(n)\}\\
&x^3_{\lfloor \frac{K}{m} \rfloor m-m}(n) = \min\{x_{\lfloor \frac{K}{m} \rfloor m-m+1}(n), x_{\lfloor \frac{K}{m} \rfloor m-m+2}(n), \ldots, x_{\lfloor \frac{K}{m} \rfloor m}(n)\}\\
&\ldots\\
&x^3_{\lfloor \frac{K}{m} \rfloor m}(n) = \min\{x_{\lfloor \frac{K}{m} \rfloor m-m+1}(n), x_{\lfloor \frac{K}{m} \rfloor m-m+2}(n), \ldots, x_{\lfloor \frac{K}{m} \rfloor m}(n)\}.
\end{aligned}
$$

In this way we obtained the new representation of the data series. The data series can now be described by the upper and the lower envelopes, or by both kinds of the envelopes, the envelopes concept being visualized in Fig. 1.

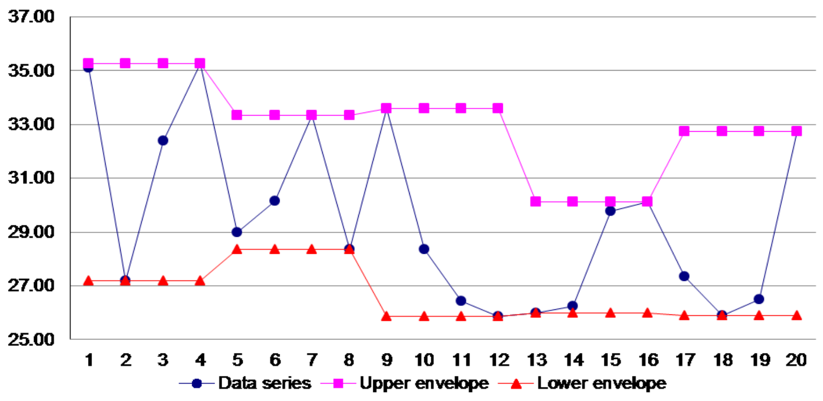


Figure 1. *4*-step upper and lower envelopes for the first 20 values of an exemplary data series

For m-step upper envelopes, i.e.

$$\{x_k^2(n)\}_{k=1}^{k=\lfloor\frac{K}{m}\rfloor m} = \left[x_1^2(n), x_2^2(n), \ldots, x^2_{\lfloor\frac{K}{m}\rfloor m}(n)\right],$$

for $n = 1, 2, \ldots, N$, m succeeding equal values are treated as a single value. So, we can replace each m sequential equal values of an envelope with a single value. In this way the dimension of such aggregated envelopes is reduced m times. The aggregated upper envelopes yield a new data series representations formally represented as:

$$\{x_k^G(n)\}_{k=1}^{k=\lfloor\frac{K}{m}\rfloor} = \left[x_1^G(n), x_2^G(n), \ldots, x^G_{\lfloor\frac{K}{m}\rfloor}(n)\right] \tag{4}$$

where

$$x^G_{1}(n) = \max\{x_{1}(n), x_{2}(n), \ldots, x_{m}(n)\},$$
$$x^G_{2}(n) = \max\{x_{m+1}(n), x_{m+2}(n), \ldots, x_{2m}(n)\},$$
$$\ldots$$
$$x^G_{\lfloor\frac{K}{m}\rfloor}(n) = \max\{x_{\lfloor\frac{K}{m}\rfloor m-m+1}(n), \ldots, x_{\lfloor\frac{K}{m}\rfloor m}(n)\}.$$

In the case of the aggregated m-step lower envelopes denoted by

$$\{x_k^D(n)\}_{k=1}^{k=\lfloor\frac{K}{m}\rfloor} = \left[x_1^D(n), x_2^D(n), \ldots, x^D_{\lfloor\frac{K}{m}\rfloor}(n)\right] \tag{5}$$

the procedure is similar. The aggregated *4*-step envelopes from Fig. 1 are visualized in Fig. 2.

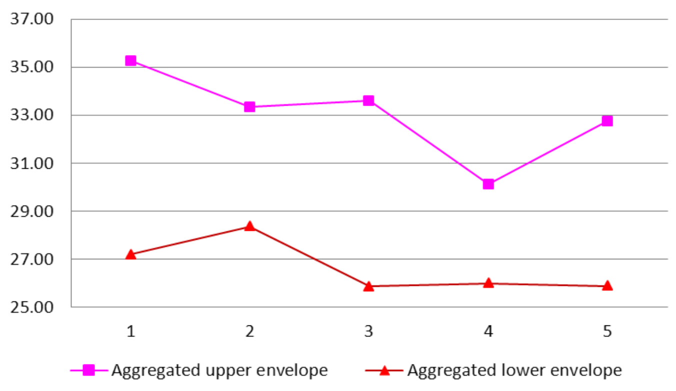


Figure 2. The aggregated *4*-step envelopes (see Fig. 1)

Due to introduction of the aggregated envelopes the dimension of the original data series can be significantly decreased. For example, for a given data series of the length K we obtain the reduced length $\lfloor\frac{K}{m}\rfloor$, where $m << K$.

3. Generation of the essential attributes

In this section we introduce another way of compressing the information of data series involved in aggregated upper or lower envelopes.

For example, let the considered data series be represented as follows:

$$\left\{x_k^G(n)\right\}_{k=1}^{k=\lfloor\frac{K}{m}\rfloor} = \left[x_1^G(n), x_2^G(n), \ldots, x_{\lfloor\frac{K}{m}\rfloor}^G(n)\right]$$

$n = 1, 2, \ldots, N$. Next we would like to compress the considered aggregated envelopes by extracting their essential attributes, and in general, due to the number q of essential attributes, where $q << \lfloor\frac{K}{m}\rfloor$, we have another reduction of dimension of the data series

$$\left\{y_i^G(n)\right\}_{i=1}^{i=q} = \left[y_1^G(n), y_2^G(n), \ldots, y_q^G(n)\right]. \tag{6}$$

It is assumed that the vector $\left\{y_i^G(n)\right\}_{i=1}^{i=q}$ can be used to reconstruct the vector $\left\{x_k^G(n)\right\}_{k=1}^{k=\lfloor\frac{K}{m}\rfloor}$ as the following vector

$$\left\{\hat{x}_k^G(n)\right\}_{k=1}^{k=\lfloor\frac{K}{m}\rfloor} = \left[\hat{x}_1^G(n), \hat{x}_2^G(n), \ldots, \hat{x}_{\lfloor\frac{K}{m}\rfloor}^G(n)\right]$$

with an assumed accuracy.

In the case of the aggregated lower envelopes $\left\{x_k^D(n)\right\}_{k=1}^{k=\lfloor\frac{K}{m}\rfloor}$ and the aggregated upper and lower envelopes $\left\{x_k^{GD}(n)\right\}_{k=1}^{k=\lfloor\frac{K}{m}\rfloor}$ the procedure is similar.

The idea of essential attributes corresponds to the principal component analysis, well known in literature, see, e.g., Jolliffe (2002). In this paper for obtaining the essential attributes the hetero association memory implemented by feedforward neural networks was applied. The neural network used consists of three layers with one hidden layer; see Krawczak, Szkatuła (2008).

The number of inputs as well as the number of outputs of the neural network is equal to $\lfloor\frac{K}{m}\rfloor$, the number of dimension of the aggregated envelopes (4) or (5). The number of hidden neurons is equal to *q,* the number of the essential attributes. In the case of applying the aggregated upper envelopes, the network is fed by the aggregated envelopes

$$\left\{x_k^G(n)\right\}_{k=1}^{k=\lfloor\frac{K}{m}\rfloor} = \left[x_1^G(n), x_2^G(n), \ldots, x_{\lfloor\frac{K}{m}\rfloor}^G(n)\right]$$

and the outputs of the network $\left\{\hat{x}_k^G(n)\right\}_{k=1}^{k=\lfloor\frac{K}{m}\rfloor} = \left[\hat{x}_1^G(n), \hat{x}_2^G(n), \ldots, \hat{x}_{\lfloor\frac{K}{m}\rfloor}^G(n)\right]$ are compared to the reference values of the aggregated envelopes

$$\left\{x_k^G(n)\right\}_{k=1}^{k=\lfloor\frac{K}{m}\rfloor} = \left[x_1^G(n), x_2^G(n), \ldots, x_{\lfloor\frac{K}{m}\rfloor}^G(n)\right].$$

The outputs of the hidden layer neurons $\left\{y_i^G(n)\right\}_{i=1}^{i=q} = \left[y_1^G(n), y_2^G(n), \ldots, y_q^G(n)\right]$ constitute the essential attributes. The case with the aggregated lower envelopes is processed in a similar way.

After the application of the standard backpropagation algorithm (or its modifications) for learning of the neural network, the essential attributes are just the outputs of hidden neurons. In this way we obtained another representation of data series, and the length q of the new representation is additionally reduced in comparison with the length $\left\lfloor \frac{K}{m} \right\rfloor$ of the aggregated envelopes data series representation.

In order to generate rules, the real values of the essential attributes must be replaced by symbolic values. The replacement is done in such a way that the ranges of the essential attributes are divided into some number of elements.

In this paper the respective calculation was done with the method called equal width interval discretization. It involves determining the domain of observed values of an essential attribute $a_j \in A$, $j = 1, \ldots, q$, and dividing this interval into equally sized intervals. The set $V_{a_j} = \{v_{j,1}, v_{j,2}, \ldots, v_{j,L_j}\}$ is the domain of the attribute a_j, and L_j denotes the number of values of the j-th attribute. One can construct interval boundaries, i.e. cut points, in the following way:

$$\begin{aligned} p_i &= \min\{V_{a_1}, V_{a_2}, \ldots, V_{a_q}\} + i \cdot \partial, \\ \partial &= \frac{\max\{V_{a_1}, V_{a_2}, \ldots, V_{a_q}\} - \min\{V_{a_1}, V_{a_2}, \ldots, V_{a_q}\}}{P}, \end{aligned} \tag{7}$$

where $i = 1, \ldots, P-1$, and $P \in \mathrm{N}$ is a parameter prescribed by the user.

4. Extraction of decision rules

Let us suppose that we have a finite set of examples U, called the learning set, and a finite set of attributes $A = \{a_1, \ldots, a_K\}$. The set $V_{a_j} = \{v_{j,1}, v_{j,2}, \ldots, v_{j,L_j}\}$ is a domain of the attribute a_j, $j = 1, \ldots, K$, L_j denotes the number of values of the j-th attribute. $V = \bigcup_{j=1,\ldots,K} V_{a_j}$ and $f : U \times A \to V$ is a function, where $f(e^n, a_j) \in V_{a_j}$ for every $a_j \in A$ and $e^n \in U$, $n = 1, 2, \ldots, N$.

Each example $e^n \in U$ is described by K attributes $A = \{a_1, \ldots, a_K\}$ and is represented by K elementary conditions in the following manner:

$$e^n = \bigwedge_{j=1}^{K} (a_j = f(e^n, a_j)) \tag{8}$$

where $f(e^n, a_j) = v_{j,t(j,n)}$, and $v_{j,t(j,n)} \in V_{a_j}$. This notation simply means that the attribute a_j takes on the value $v_{j,t(j,n)}$ in the example e^n. The index $t(j,n)$ for $j \in \{1, 2, \ldots, K\}$ and $n \in \{1, 2, \ldots, N\}$ specifies, which value of the j-th attribute is used in the n-th example.

For instance, for the attributes: *height*, *colour of hair*, *colour of eyes*, we can describe the aspect of a person number 1 as

$$(\textit{height} = \text{‘high’}) \wedge (\textit{color of hair} = \text{‘blond’}) \wedge (\textit{color of eyes} = \text{‘blue’}).$$

An example e^n is composed of K ‘attribute-value’ pairs, denoted $s_j = (a_j = v_{j,t(j,n)})$. The conjunction of l ‘attribute-value’ pairs, $l \leq K$, i.e. $\bigwedge_{j \in I} s_j = C^I$, where $I \subseteq \{1, \ldots, K\}$, $card(I) = l$, is called a *complex.*

We say that the complex C^I *covers* an example e^n if all the conditions of attributes given as j-th selectors are covered by (equal to) the values of the respective attributes in the example, $\forall j \in I$. The set of all the examples described by the conjunction C^I is denoted $[C^I]$. For instance, the complex ($\textit{color of hair} = \text{‘blond’}$) covers the example

$$(\textit{height} = \text{‘high’}) \wedge (\textit{color of hair} = \text{‘blond’}) \wedge (\textit{color of eyes} = \text{‘blue’}).$$

Suppose that we have *a decision attribute* a_d, where $\{a_d\} \bigcap A = \emptyset$ and $V_{a_d} = \{v_{d,1}, v_{d,2}, \ldots, v_{d,L_d}\}$ is the domain of the attribute a_d. We can perform the partition of the entire set of examples into the disjoint classes with respect to the values taken by this attribute. The elements of the set A are referred to as *conditional attributes.* We assume that the number and character of attributes are sufficient for the correct split of examples belonging to different classes.

We have the sets $\{U_{v_{d,l}} : l = 1, \ldots, L_d\}$, where

$$U_{v_{d,l}} = \{e \in U : f(e, a_d) = v_{d,l}\}, \forall\, v_{d,l} \in V_{a_d}, \tag{9}$$

where

$$U_{v_{d,1}} \cup \ldots \cup U_{v_{d,L_d}} = U, \quad U_{v_{d,i}} \cap U_{v_{d,j}} = \emptyset \text{ for } i \neq j.$$

The decision attribute splits the set of examples into the non-empty, disjoint and exhaustive subsets that we call *decision classes.*

The sets of the learning examples determined in this manner along with their division into classes, are the starting point in the process of machine learning, which is supposed to lead to the descriptions of the classes considered. The process of formation of a class description on the basis of the set of examples having certain common properties, which distinguish a given class from the others, is characterized by the adopted language of data representation and the applied algorithm of machine learning.

These descriptions of the classes can be represented either in the form of elementary rules being the logical expressions of the form IF *certain conditions are fulfilled* THEN *membership in a definite class takes place*; in the form of decision trees; or in the form of the appropriately selected connection weights in neural networks and their structure.

In this paper the belonging of data series to proper class is modelled in the form of rules, the rules give a prescription of class representations. In our case, the conditional part of the rules will contain the conjunction of conditions related to the subset of attributes selected for the description of the examples.

An implication

$$R_k : \text{IF } C^{I_k} \text{ THEN } (a_d = v_{d,l}), \tag{10}$$

$l \in \{1, \ldots, L_d\}$, is called the *k-th elementary rule* for class $U_{v_{d,l}}$, where $C^{I_k} = \bigwedge_{j \in I_k} [a_j = v_{j,t(j,k)}]$ is description of example in terms of condition attributes a_j, $j \in I_k$, $I_k \subseteq \{1, \ldots, K\}$ and this example belongs to class $U_{v_{d,l}}$. The index $t(j,k)$ specifies which value of the j-th attribute is used in the k-th rule.

Each rule is characterized by the coefficient of its strength. The strength of the rule R_k, which depends upon the number of examples, described by the conditional part of the rule C^{I_k}, belonging to a given class $U_{v_{d,l}}$ is defined in the following manner:

$$s(C^{I_k}) \;=\; \frac{card(\{e : e \in [C^{I_k}] \text{ and } f(e, a_d) = v_{d,l}\})}{card(\{e : e \in U_{v_{d,l}}\})} \tag{11}$$

It is evident that $0 \leq s(C^{I_k}) \leq 1$. The more examples are described by the rule, the bigger the rule strength coefficient (i.e. the rule is more important).

For instance, for the problem considered in Experiment 1, the elementary rule

$$R_1 : \text{ IF } \quad (a_5 = 9) \quad \text{THEN (Class} = 1)$$

covers 19 examples and $card(U_{v_{d,l}}) = 25$. Then, using (11), we obtain $s(C^{I_1}) = 19/25 = 0.76$.

A set of rules will be called *complete*, if it is non-empty and finite and if $\forall e \in U$ there exists at least one rule describing it. A set of rules which is complete and contains the minimum number of rules is called the *minimal set of rules*.

The rules, mentioned above, can be formed by applying various algorithms of machine learning from examples. It is possible to establish:

- the minimal set of rules,
- the complete set of all minimal rules,
- or the set of strong rules.

In view of our experience, the problem was calculated with the method developed by Szkatuła (1995, 2002), Kacprzyk and Szkatuła (1999, 2002, 2005a,b, 2010), which creates the minimal set of rules successively for each class. The rules must guarantee fulfillment of some requirements, namely they should describe 'all' or 'almost all' examples which belong to the considered class, and they should not describe 'all' or 'almost all' examples which do not belong to the considered class. Additionally, they should have minimal length in the sense of a number of conditions, and so forth.

The set of examples is represented in our case as the table in the computer program, where each example and each attribute is described by a unique value taken by the attribute in the example. In the task considered we deal with nominal non-ordered attributes.

In our case, the conditional part of the rules contained the conjunction of conditions related to the subset of the essential attributes. The process of generating the decision rules is based on a set of examples under the assumption that for each class the examples have some common properties which distinguish them from another class.

Example 1 Assume that we have two classes $U_{v_{d,1}}$ and $U_{v_{d,2}}$. Suppose that in Fig. 3 all the learning examples are shown. Those belonging to class $U_{v_{d,1}}$ are marked with $\oplus$ and those belonging to class $U_{v_{d,2}}$ are marked with Θ.

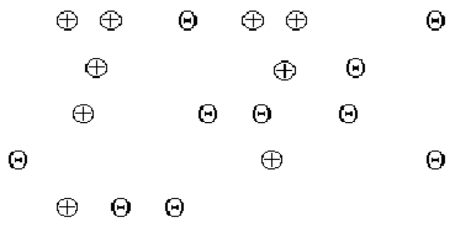


Figure 3. The learning examples

Consider class $U_{v_{d,1}}$. First, we can find the set of four elementary rules for class $U_{v_{d,1}}$, correctly describing all the training examples, see Fig. 4.

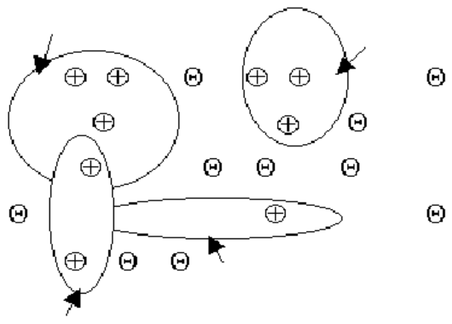


Figure 4. Four elementary rules for class $U_{v_{d,1}}$

Next, we can determine in the same way the set of the elementary rules for class $U_{v_{d,2}}$, see Fig. 5. The set of elementary rules for two classes is illustrated in Fig. 6.

The rules formed in this manner can be applied to classification of new examples, the ones that have not appeared in the learning process.

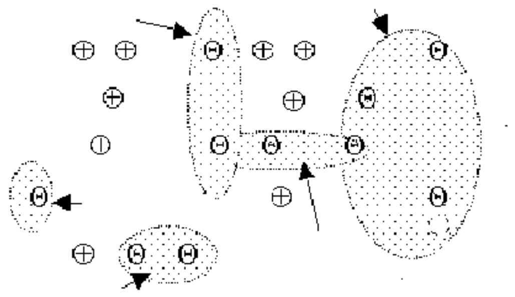


Figure 5. Five elementary rules for class $U_{v_{d,2}}$

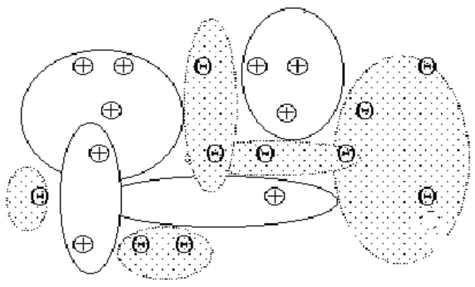


Figure 6. The set of elementary rules for two classes

5. Basic elements of the proposed hybrid system

This paper describes a hybrid system for solving a classification problem of multidimensional data series. We considered the set of learning examples, i.e. data series $\{x_k(n)\}_{k=1}^{k=K}$ for $n = 1, 2, \ldots, N$, with their division into classes (Krawczak, Szkatuła, 2008). The system consists of five main modules:

Data Normalization Module

The module is responsible for standardisation of each considered data series so that the average is equal zero and the standard deviation is equal one.

Aggregation Envelope Module

The m-step upper envelopes, i.e. $\left\{x_k^2(n)\right\}_{k=1}^{k=\left\lfloor \frac{K}{m} \right\rfloor m}$ and/or m-step lower envelopes are established, i.e. $\left\{x_k^3(n)\right\}_{k=1}^{k=\left\lfloor \frac{K}{m} \right\rfloor m}$, for $m << K$(Krawczak, Szkatuła, 2010a,b).

The envelopes are aggregated, i.e. $\left\{x_k^G(n)\right\}_{k=1}^{k=\left\lfloor \frac{K}{m} \right\rfloor}$ and/or $\left\{x_k^D(n)\right\}_{k=1}^{k=\left\lfloor \frac{K}{m} \right\rfloor}$ (Krawczak, Szkatuła, 2010a,b).

Essential Attributes Generation Module

The essential attributes are generated, i.e. $\left\{y_i^G(n)\right\}_{i=1}^{i=q}$ and/or $\left\{y_i^D(n)\right\}_{i=1}^{i=q}$ and/or $\left\{y_i^{GD}(n)\right\}_{i=1}^{i=q}$ for $q << \left\lfloor \frac{K}{m} \right\rfloor$ (Krawczak, Szkatuła, 2010a,b; Krawczak, 2003a,b).

Essential Attributes Nominalization Module

The real values of the essential attributes are replaced by the symbolic values (Krawczak, Szkatuła, 2010a,b).

Rule Generation Module

After obtaining the symbolic values of the essential attributes we can start generating the rules, i.e. (Szkatuła, 1995, 2002; Szkatuła, Kacprzyk, 2005; Krawczak, Szkatuła, 2010a,b).

The rules were verified by using the learning and testing data series. The ratio of correct classification decisions to the total number of decisions made was taken as the measure of classification accuracy, in percentage.

Such classification is carried out through verification of fulfillment of conditions in the conditional parts of the rules, and in case of equivocal situations (when more than one, or none, of the rules is fulfilled), the degree of matching of the class is calculated (Szkatuła, 1995, 2002).

6. Experimental results

In order to verify the proposed hybrid system concept the database available at the Irvine University of California was used (Alcock, Manolopoulos, 1999), http://kdd.ics.uci.edu/databases/synthetic_control/synthetic_control.data.html.

The database consists of the data series synthetically generated by the equations. Each equation represents a different type of pattern. Each pattern was taken as a time series of 60 data points. The following equations were used to create the data points $z(t)$, where $1 \leq t \leq 60$, for the various patterns:

E: $z(t) = v + rs + kx$

F: $z(t) = v + rs - kx$

A: $z(t) = v + rs$,

where, for each pattern, v is the mean value of the process variable under observation ($v = 80$), s is the standard deviation of the process variable ($s = 5$), r is a random number between -3 and 3, x is the magnitude of the shift (x takes a value between 7.5 and 20), k indicates the shift position in E and F ($k = 0$ before the shift and $k = 1$ at the shift and thereafter).

Our aim is to generate a set of elementary rules for the three considered classes:

- *Class 1*: the data belongs to series E,
- *Class 2*: the data belongs to series F,
- *Class 3*: the data belongs to series A,

that could be used for classification of other data series, not classified before.

We considered the following learning data series: the first 25 belong to Class 1, the next 25 belong to Class 2 and the rest 25 belong to Class 3. Each data series has 60 values, as follows $\{x_k(n)\}_{k=1}^{k=60} = [x_1(n), x_2(n), \ldots, x_{60}(n)]$, $n = 1, 2, \ldots, N$.

The considered normalized learning data series are shown in Fig. 7. Table 1 shows three selected examples of the considered normalized data series.

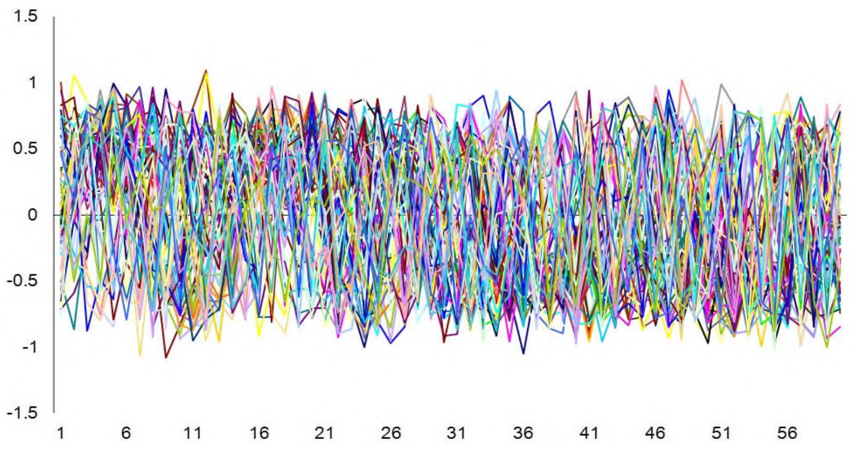


Figure 7. The image shows (after normalization): 25 data series of Class 1, 25 data series of Class 2 and 25 data series of Class 3

Table 1.

$x_1(15)$	$x_2(15)$	$x_3(15)$	$\cdots$	$x_{58}(15)$	$x_{59}(15)$	$x_{60}(15)$
0.37	0.80	0.23	...	0.29	0.48	0.52

$x_1(21)$	$x_2(21)$	$x_3(21)$	$\cdots$	$x_{58}(21)$	$x_{59}(21)$	$x_{60}(21)$
-0.03	0.35	0.55	...	0.71	0.39	0.,24

$x_1(22)$	$x_2(22)$	$x_3(22)$	$\cdots$	$x_{58}(22)$	$x_{59}(22)$	$x_{60}(22)$
0.01	-0.04	0.50	...	-0.36	0.00	0.21

In the paper three following computational experiments are considered:

Experiment 1. The aggregated *4*-step upper envelopes were used to create five essential attributes.

Experiment 2. The aggregated *4*-step lower envelopes were used to create five essential attributes.

Experiment 3. Both the aggregated *4*-step upper and aggregated *4*-step lower envelopes were used to create five essential attributes.

Thus the original data series were represented by five essentials attributes and can be treated as learning data for generating elementary rules for the three considered classes. The results are shown below.

Experiment 1

Our first goal was to reduce the dimensionality of the normalized data. For each considered data series the *4*-step upper envelopes of the learning data series $[x_1(n), x_2(n), \ldots, x_{60}(n)]$ belonging to Class 1, Class 2 and Class 3 were calculated and aggregated, we obtained $[x_1^G(n), x_2^G(n), \ldots, x_{15}^G(n)]$, for $n = 1, 2, \ldots, 75$, see Fig. 8. For the examples from Table 1 the aggregated *4*-step upper envelopes look like in Table 2.

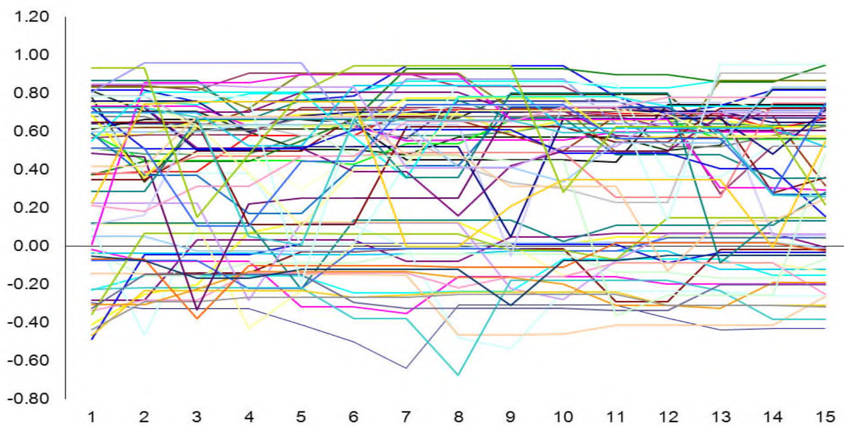


Figure 8. The aggregated 4-step upper envelopes of data series belonging to Class 1, Class 2 and Class 3

Table 2.

$x_1^G(15)$	$x_2^G(15)$	$x_3^G(15)$	$\cdots$	$x_{15}^G(15)$
0.64	0.91	-0.01	…	0.53

$x_1^G(21)$	$x_2^G(21)$	$x_3^G(21)$	$\cdots$	$x_{15}^G(21)$
0.55	0.77	-0.52	…	0.71

$x_1^G(22)$	$x_2^G(22)$	$x_3^G(22)$	$\cdots$	$x_{15}^G(22)$
0.50	0.98	-0.23	…	0.21

In order to find the essential attributes of the aggregated upper envelopes a three layer feedforward neural network was applied with different numbers of neurons within the hidden layer. All neurons have sigmoidal activation functions.

Several computational experiments were performed in order to find the optimal number of hidden neurons, and the results of the experiment are shown in Fig. 9.

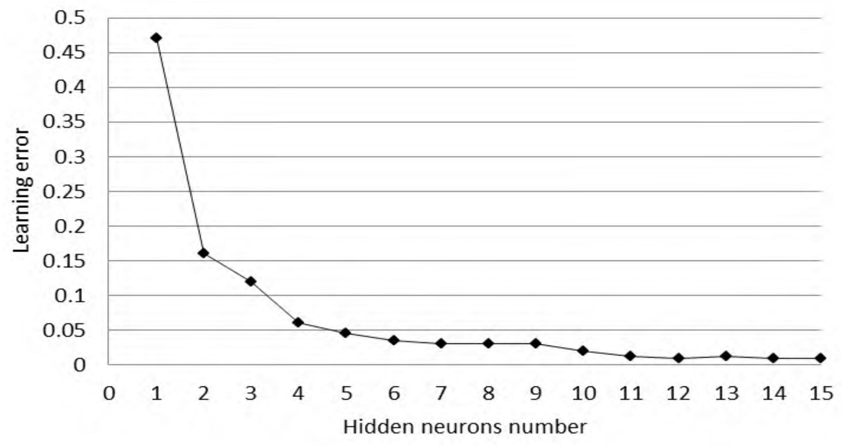


Figure 9. Values of learning error vs. the number of hidden neurons

According to the experiment the number of neurons of the hidden layer was chosen as 5, meaning that we assumed that five essential attributes conserve the information about the data series character. For the experiment, as well for the learning of the neural network, the backpropagation algorithm was used.

In the next step of our procedure we normalized the data describing the essential attributes, and the outputs of the hidden layer were multiplied by 1000.

The exemplary values of the essential attributes for the examples number 15, 21 and 22 are shown in Table 3.

Table 3.

$y_1^G(15)$	$y_2^G(15)$	$y_3^G(15)$	$y_4^G(15)$	$y_5^G(15)$
46	354	46	736	866

$y_1^G(21)$	$y_2^G(21)$	$y_3^G(21)$	$y_4^G(21)$	$y_5^G(21)$
55	495	43	428	843

$y_1^G(22)$	$y_2^G(22)$	$y_3^G(22)$	$y_4^G(22)$	$y_5^G(22)$
46	535	60	485	740

Thus, the original data series can be represented by a set of five essentials attributes $\{y_1^G(n), y_2^G(n), \ldots, y_5^G(n)\}$, see Fig. 10.

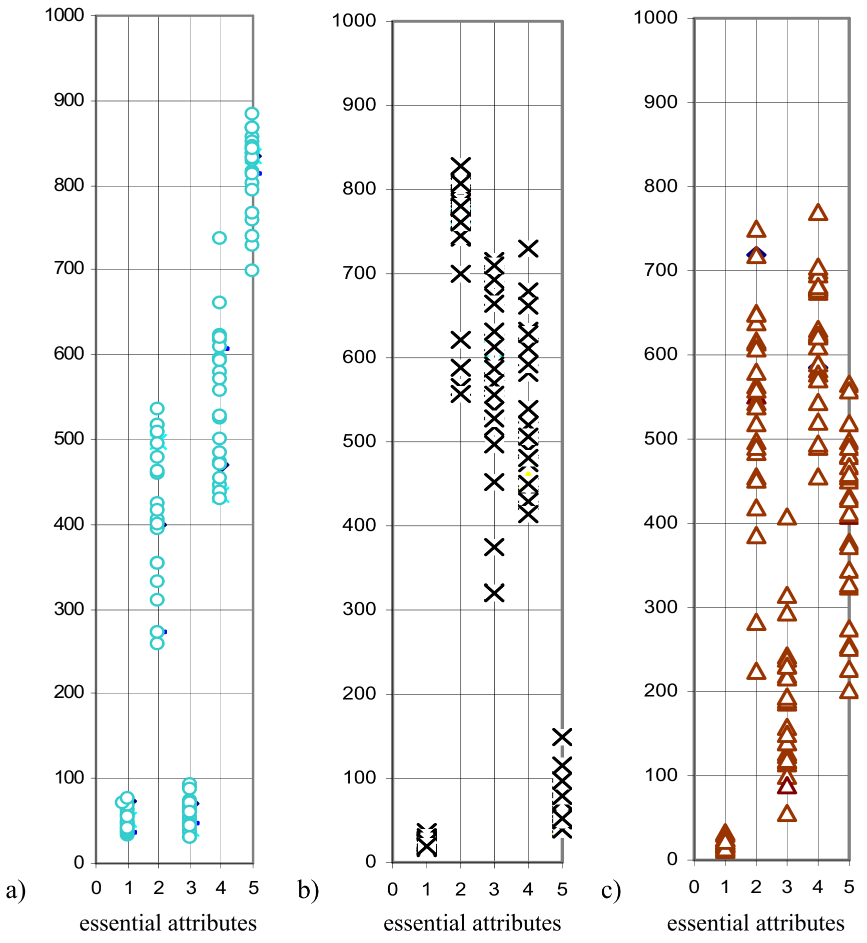


Figure 10. The values of the five essential attributes describing the learning data series belong to Class 1 (a), Class 2 (b) and Class 3 (c)

It is worth mentioning that in Fig. 10 there are plots of the five essential attributes for some chosen perturbation of the numbers {1, 2, 3, 4, 5}, in this case the attributes form a vector. For each class the shapes of the plots are meaningfully different, it means that the composition of the essential attributes in some sense describes (remembers) the class of each data series.

For a different perturbation of the set numbers of the essential attributes we can observe the same effect.

Table 4. The nominalization of the essential attributes

Essential attributes value section	Nominal value
>= 0 and <= 100	*a*
> 100 and <= 200	*b*
> 200 and <= 300	*c*
> 300 and <= 400	*d*
> 400 and <= 500	*e*
> 500 and <= 600	*f*
> 600 and <= 700	*g*
> 700 and <= 800	*h*
> 800 and <= 900	*i*
> 900 and <= 1000	*j*

Table 5.

$y_1^G(15)$	$y_2^G(15)$	$y_3^G(15)$	$y_4^G(15)$	$y_5^G(15)$
a	*d*	*a*	*h*	*i*

$y_1^G(21)$	$y_2^G(21)$	$y_3^G(21)$	$y_4^G(21)$	$y_5^G(21)$
a	*e*	*a*	*e*	*i*

$y_1^G(22)$	$y_2^G(22)$	$y_3^G(22)$	$y_4^G(22)$	$y_5^G(22)$
a	*f*	*a*	*e*	*h*

Next, the nominalization of the essential attributes was arranged in the way shown in Table 4. In results the values of the essential attributes take one of the nominal symbolic values: a, b, c, d, e, f, g, h, i, j.

The exemplary values of the essential attributes after the nominalization for the examples number 15, 21 and 22 are shown in Table 5.

Thus, each example is represented by values of the five essential attributes and can now be treated as learning data for generation of elementary rules in the following form

IF *some conditions are satisfied* THEN *the data belongs to a proper class.*

In our case, the conditional part of the rules will contain the conjunction of conditions related to the subset of the essential attributes. The classification accuracy of the rules derived is the percentage of examples correctly classified.

The minimal set of rules, R_k, $k = 1, \ldots, 9$, obtained is shown in Table 6; the strength s of the rule (11) and the number of examples described by the rule is given for each case.

The rules correctly classified 100% of the learning data series belonging to Class 1, Class 2 and Class 3.

Table 6.

R_k	The strength of the rule	The number of described learning examples
R_1 : IF ($a_5 = i$) THEN (Class =1)	0.76	19
R_2 : IF ($a_5 = h$) THEN (Class =1)	0.20	5
R_3 : IF ($a_5 = g$) THEN (Class =1)	0.04	1
R_4 : IF ($a_5 = a$) THEN (Class =2)	0.92	23
R_5 : IF ($a_5 = b$) THEN (Class =2)	0.08	2
R_6 : IF ($a_3 = b$) THEN (Class =3)	0.52	13
R_7 : IF ($a_5 = e$) THEN (Class =3)	0.48	12
R_8 : IF ($a_3 = c$) THEN (Class =3)	0.28	7
R_9 : IF ($a_5 = d$) THEN (Class =3)	0.20	5

Table 7.

$\cdots$	$\cdots$	$y_3^G(15)$	$\cdots$	$y_5^G(15)$
…	…	a	…	i

$\cdots$	$\cdots$	$y_3^G(21)$	$\cdots$	$y_5^G(21)$
…	…	a	…	i

When examining Table 6 we can notice that only two out of five essential attributes participate in creating the generated rules, and these two attributes, a_3 and a_5, are sufficient to properly classify the data series. It is worth noticing that in the space of these two attributes there are several examples overlapping, e.g. example 15 and 21 are not distinguishable, as shown in Table 7.

All of the considered examples, used in the learning process, which belong to Classes 1, 2 or 3, and described solely by the two attributes a_3 and a_5 in the generated rules are depicted in Fig. 11.

Next, the rules were verified by using 75 new testing data series which did not participate in the generation of rules (25 for each of three classes). The rules correctly classified 97.3% testing data series (i.e. 73 data series).

Experiment 2

For each considered data series the *4*-step lower envelopes of the learning data series $[x_1(n), x_2(n), \ldots, x_{60}(n)]$ belonging to Class 1, Class 2 and Class 3 were calculated and aggregated, and we obtained $[x_1^D(n), x_2^D(n), \ldots, x_{15}^D(n)]$, for $n = 1, 2, \ldots, 75$, see Fig. 12.

In order to compress the envelopes, a three layer feedforward neural network was applied. The network has 15 inputs, 15 outputs and 5 neurons within the

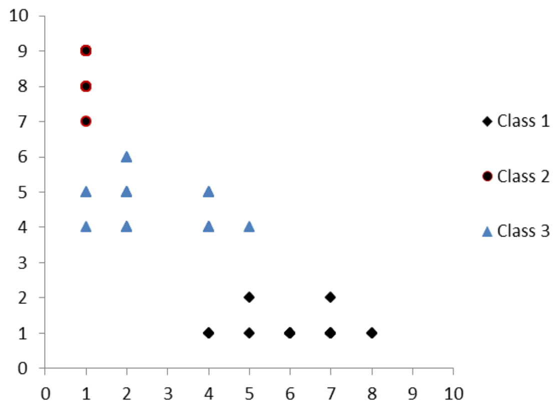


Figure 11.

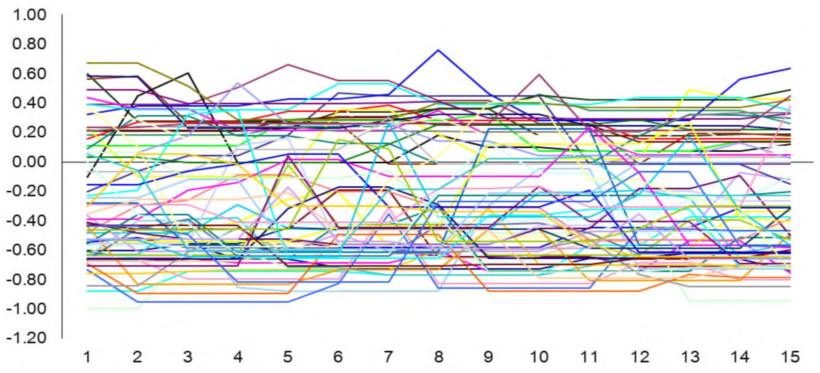


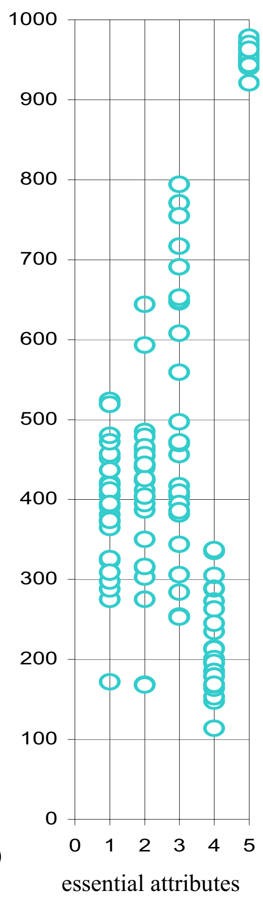
Figure 12. The aggregated 4-step lower envelopes of data series belonging to Class 1, Class 2 and Class 3

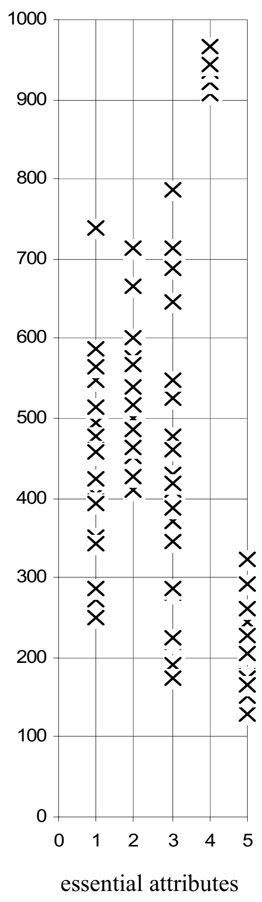
hidden layer. Thus, the original data series can be represented by set of five essential attributes $\{y_1^D(n), y_2^D(n), \ldots, y_5^D(n)\}$, see Fig. 13.

Next, the nominalization of the essential attributes was arranged in the way shown in Table 4.

The minimal set of rules R_k, $k = 1, \ldots, 5$, obtained is shown in Table 8; the strength of the rule (11) and the number of examples described examples by a rule is given for each case. The rules correctly classified 100% of the learning data series.

Next, the rules were verified by using 75 new testing data series, which did not participate in the generation of rules (25 of each of three classes). The rules correctly classified 98.7 % testing data series (i.e. 74 data series).

a)


b)


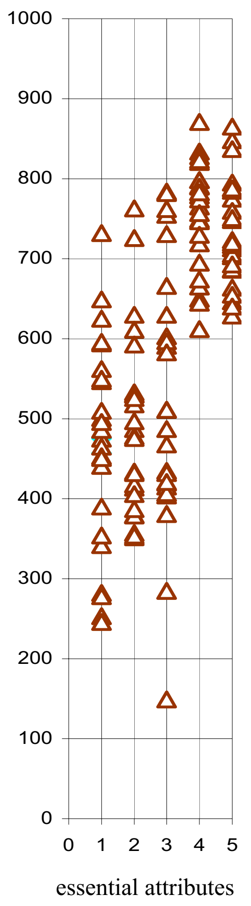
c)


Figure 13. The values of the five essential attributes describing the learning series belonging to Class 1 (a), Class 2 (b) and Class 3 (c)

Table 8.

R_k	The strength of the rule	The number of described learning examples
R_1 : IF ($a_5 = j$) THEN (Class = 1)	1.00	25
R_2 : IF ($a_4 = j$) THEN (Class = 2)	1.00	25
R_3 : IF ($a_5 = h$) THEN (Class = 3)	0.56	14
R_4 : IF ($a_5 = g$) THEN (Class = 3)	0.32	8
R_5 : IF ($a_4 = g$) THEN (Class = 3)	0.24	6

Experiment 3

Both the *4*-step upper envelopes and *4*-step lower envelopes of the learning data series $[x_1(n), x_2(n), \ldots, x_{60}(n)]$ belonging to Class 1, Class 2 and Class 3 were aggregated and we obtained $\left[x_1^G(n), x_2^G(n), \ldots, x_{15}^G(n), x_1^D(n), x_2^D(n), \ldots, x_{15}^D(n)\right]$, for $n = 1, 2, \ldots, 75$, see Fig. 14.

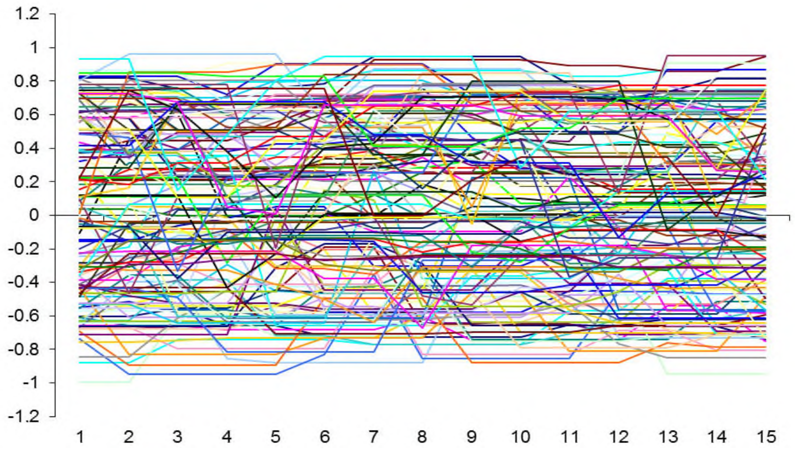


Figure 14. The aggregated 4-step upper and 4-step lower envelopes of data series

In order to compress the envelopes, a three layer feedforward neural network was applied. The network has 30 inputs, 30 outputs and 5 neurons within the hidden layer. Thus, the original data series can be represented by a set of five essential attributes $\{y_1^{GD}(n), y_2^{GD}(n), \ldots, y_5^{GD}(n)\}$, see Fig. 15.

Next, the nominalization of the essential attributes was arranged in the way shown in Table 4.

The obtained minimal set of rules R_k, $k = 1, \ldots, 17$, the strength s of the rule (11) and the number of described examples are shown in Table 9. The rules correctly classified 100% of the learning data series.

Next, the rules were verified by using 75 new testing data series which did not participate in the generation of rules (25 of each of three classes). The rules correctly classified 93.3 % of testing data series (i.e. 70 data series).

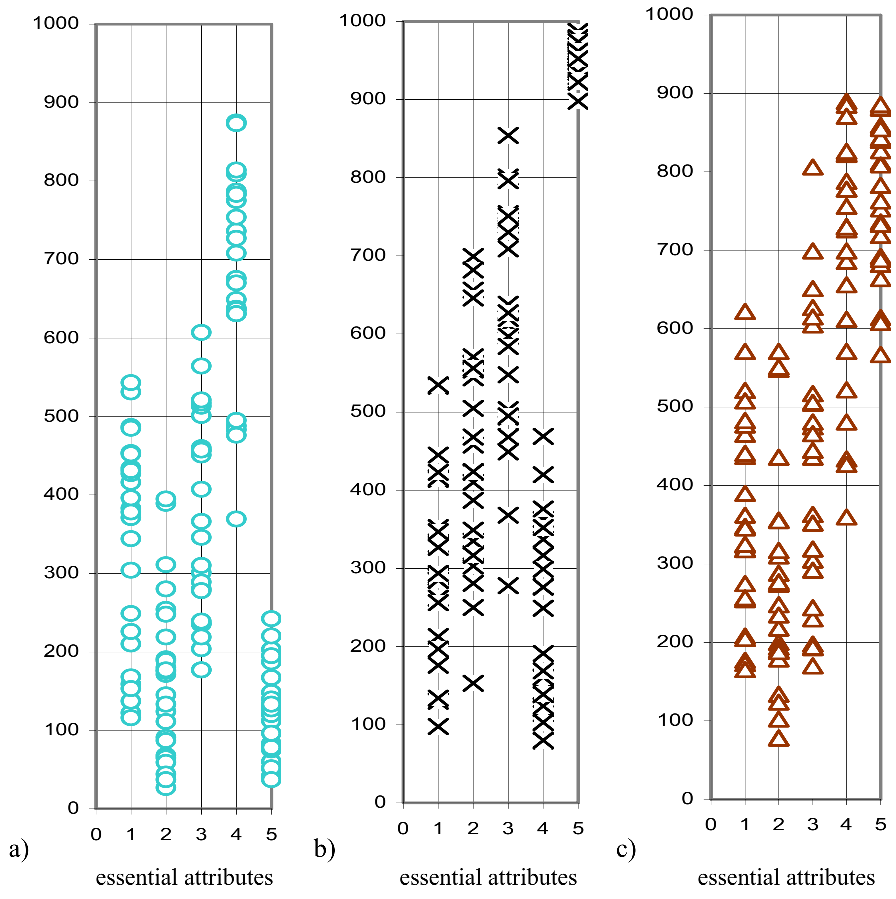


Figure 15. The values of the five essential attributes describing the learning data series belonging to Class 1 (a), Class 2 (b) and Class 3 (c)

Table 9.

R_k	The strength of the rule	The number of described learning examples
R_1 : IF ($a_5 = b$) THEN (Class = 1)	0.72	18
R_2 : IF ($a_5 = c$) THEN (Class = 1)	0.16	4
R_3 : IF ($a_5 = a$) THEN (Class = 1)	0.12	3
R_4 : IF ($a_4 = b$) THEN (Class = 2)	0.44	11
R_5 : IF ($a_4 = c$) THEN (Class = 2)	0.20	5
R_6 : IF ($a_4 = a$) THEN (Class = 2)	0.16	4
R_7 : IF ($a_4 = d$) THEN (Class = 2)	0.16	4
R_8 : IF ($a_3 = i$)$\wedge$($a_4 = e$) THEN (Class = 2)	0.04	1
R_9 : IF ($a_5 = g$) THEN (Class = 3)	0.36	9
R_{10} : IF ($a_4 = g$) THEN (Class = 3)	0.32	8
R_{11} : IF ($a_5 = h$) THEN (Class = 3)	0.28	7
R_{12} : IF ($a_4 = i$)$\wedge$($a_5 = i$) THEN (Class = 3)	0.08	2
R_{13} : IF ($a_1 = a$) THEN (Class = 3)	0.08	2
R_{14} : IF ($a_2 = b$)$\wedge$($a_5 = i$) THEN (Class = 3)	0.04	1
R_{15} : IF ($a_5 = f$) THEN (Class = 3)	0.04	1
R_{16} : IF ($a_3 = h$)$\wedge$($a_5 = i$) THEN (Class = 3)	0.04	1
R_{17} : IF ($a_1 = d$)$\wedge$($a_2 = h$) THEN (Class = 3)	0.04	1

7. Conclusions

In this paper we introduced a new hybrid concept of data series representation for solving the classification problems. The concept is first based on the so called 'upper and lower envelopes' and 'aggregation of the envelopes', and then on essential attributes of the envelopes. Both representations allow for the reduction of dimensionality of the original data series.

Next we generated a set of rules to classify the exemplary data series. The original data must be prepared in an accessible way for obtaining the envelopes as well as the normalization and the nominalization for obtaining the rules.

A numerical example shows that even after a reduction of dimensionality (as well as reduction of information), the new representations preserves information about the data series characteristics.

The tests show that the accuracy of the hybrid methodology is similar for the upper and lower data series envelopes, and it is sufficient. In the case of using both upper and lower envelopes the accuracy obtained is rather not satisfying, the reason being that the number of essential attributes was assumed too small.

References

Alcock, R.J. and Manolopoulos, Y. (1999) Time-Series Similarity Queries Employing a Feature-Based Approach. *7th Hellenic Conference on Informatics*, Ioannina, Greece.

Benedikt, L., Kajic, V., Cosker, D., Marshall, D. and Rosin, P.L. (2008)Facial Dynamics in Biometric Identification. *Proc. of British Machine Vision Conference*, Leeds, 2008.

Chan K. and Fu A.W. (2005) Efficient time series matching by wavelets. *Proceedings 15th IEEE International Conference on Data Engineering.* Sydney, Australia, 126-133.

Choy, E., Krawczak, M., Shannon, A. and Szmidt, E., eds. (2007) *A Survey of Generalized Nets.* KvB Institute of Technology, Sydney, Australia.

Faloutsos, C., Ranganathan, M. and Manolopulos, Y. (1994) Fast subsequence matching in time-series databases. *SIGMOD Record*, 23, 519-529.

Jolliffe, I.T. (2002) *Principal Component Analysis.* Springer.

Kacprzyk, J. and Szkatuła, G. (1999) An inductive learning algorithm with a preanalysis data. *International Journal of Knowledge - Based Intelligent Engineering Systems*, **3**, 135-146.

Kacprzyk, J. and Szkatuła, G. (2002) An integer programming approach to inductive learning using genetic and greedy algorithms. In: L.C. Jain and J.Kacprzyk, eds., *New Learning Paradigms in Soft Computing. Studies in Fuzziness and Soft Computing.* Physica-Verlag Heidelberg, 323-367.

Kacprzyk, J. and Szkatuła, G. (2005a) A softened formulation of inductive learning and its use for coronary disease data. *Lecture Notes in Artificial Intelligence*, **3488**, 200-209.

Kacprzyk, J. and Szkatuła, G. (2005b) An inductive learning algorithm with a partial completeness and consistence via a modified set covering problem. *Lecture Notes in Computer Science*, **3697**, 661-666.

Kacprzyk, J. and Szkatuła, G. (2010) Inductive Learning: A Combinatorial Optimization. In: J. Koronacki, Z.W. Ras, S.T. Wierzchoń, J. Kacprzyk, eds., *Advances in Machine Learning. Studies in Computational Intelligence*, **262**, Springer.

Keogh, E., Chakrabarti, K. and Pazzani, M. (2001) Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. In: *Proc. of ACM SIGMOD Conference on Management of Data. Santa Barbara. May 21-24*, 151-162.

Krawczak, M. (2003a) *Multilayer Neural Systems and Generalized Net Models.* Ac. Publ. House EXIT, Warsaw.

Krawczak, M. (2003b) Heuristic dynamic programming - Learning as control problem. In: L. Rutkowski, J. Kacprzyk, eds., *Neural Networks and Soft Computing.* Physica Verlag, Heidelberg, 218-223.

Krawczak, M. (2006) A novel modelling methodology: generalized nets. In: A. Cader, L. Rutkowski, R. Tadeusiewicz, J. Żurada, eds., *Artificial In-*

telligence and Soft Computing. Ac. Publ. House EXIT, Warsaw.

Krawczak, M. and Szkatuła, G. (2008) On decision rules application to time series classification. In: K.T. Atanassov et al., eds., *Advances in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics*, Ac. Publ. House EXIT.

Krawczak, M., Szkatuła, G. (2010a) Time series envelopes for classification. *IEEE Intelligent Systems Conference*, London, July 7-9, 2010.

Krawczak, M. and Szkatuła, G. (2010b) On time series envelopes for classification problems. In: K.T. Atanassov et al., eds., *Developments in Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics.* SRI PAS, Warsaw.

Krawczak, M. and Szkatuła, G. (2010c) On time series clustering. *Technical Report.* Systems Research Institute PAS, Warsaw.

Lin, J., Keogh, E., Patel, P. and Lonardi, S. (2002) Finding motifs in time series. *The 2nd Workshop on Temporal Data Mining, the 8th ACM International Conference on Knowledge Discovery and Data Mining.* Edmonton, Canada, 53-68.

Nanopoulos, A., Alcock, R. and Manolopoulos, Y. (2001) Feature-based Classification of Time-series Data. *International Journal of Computer Research*, 49-61.

Rodríguez, J.J. and Alonso, C.J. (2004) Interval and dynamic time warping-based decision trees. *Proceedings of the 2004 ACM symposium on applied computing* (SAC), 548-552.

Shahabi, C., Tian, X. and Zhao, W. (2000) TSA-tree: A wavelet-based approach to improve the efficiency of multi-level surprise and trend queries. *Proceedings of the 12th International Conference on Scientific and Statistical Database Management.* Berlin, 55-68.

Szkatuła, G. (1995) *Machine learning from examples under errors in data*, Ph.D. Thesis, SRI PAS Warsaw, Poland.

Szkatuła, G. (2002) Application of modified covering problem in machine learning. In: J. Gutenbaum, ed., *Automatics Control Management.* SRI PAS, Warsaw, 431-445.

Szkatuła, G. and Kacprzyk, J. (2005) An inductive learning algorithm with a partial completeness and consistency. In: M. Dramiński, P. Grzegorzewski, T. Trojanowski, S. Zadrożny, eds., *Issues in intelligent systems. Models and techniques.* EXIT, Warszawa, 229-246.

Wu, Y. and Chang, E.Y. (2004) *Distance-function design and fusion for sequence data.* CIKM '04, 324-333.

Xi, X., Keogh, E.J., Shelton, C.R. and Wei, L. and Ratanamahatana, C.A. (2006)*Fast time series classification using numerosity reduction. ICML*, 2006.