Design Methodology for Rough-neuro-fuzzy Classification with Missing Data

Robert K. Nowicki *Member, IEEE*
Marcin Korytkowski
Rafał Scherer *Member, IEEE*
Institute of Computational Intelligence
Czestochowa University of Technology
ul. Armii Krajowej 36, 42-200 Czestochowa, Poland
Email: {robert.nowicki, rafal.scherer}@iisi.pcz.pl

Bartosz A. Nowak
Department of Mathematical Methods in Computer Science
University of Warmia and Mazury
ul. Słoneczna 54
10-710 Olsztyn, Poland
Email: bnowak@matman.uwm.edu.pl

***Abstract*—One of important methods designed to classify objects with missing feature values are rough neuro–fuzzy classifiers (RNFC). Similarly to neuro–fuzzy systems, they are specific network structures, which can be trained by optimization methods based on gradient descent. However, to the best of our knowledge, there are no publications concerning such way of RNFC designing. In the paper, problems with gradient learning of RNFC are identified and suitable solutions are proposed. The influence of missing values level on the learning process and classification quality is examined. The RNFC is compared with the k-NN classifier adapted to missing values problem by a "wide imputation" method. All experiments use 10-fold cross validation.**

I. Introduction

The problem of missing data is inherent in practical applications of classifiers, controllers and approximators. Hence, many methods solving this issue have been proposed. Moreover, the causes of information inaccessibility have been analysed and classified, thus specific methods are proper for particular cases. Generally, there are two approaches to process data with missing values. In the first one, input data with missing values must be subjected to preprocessing. Then, any of known methods devoted to tasks with complete input information, such as neural networks, fuzzy systems, k-nn classifiers etc. can be applied. Most common types of preliminary conversions are:

- Marginalisation — importance of data that do not contain values are reduced, usually by simply deletion. The most common methods of reduction are:
  - List-wise deletion — samples with a missing values are removed.
  - Attribute deletion — attributes that have lost values are deleted.
  - Pair-wise deletion — samples that have missing values in examined attributes for chosen analysis are ignored. A set of not ignored samples may be different for every analysis, because everyone can work on different attributes.
- Imputation — places with no data are filled by approximated ones. A variety of methods is wide, e.g.
  - Average, median — missing values of the attribute are filled by average or median of known values of the same attribute from all available samples.
  - Random — missing values of the attribute are filled by random values from the domain of the attribute. The specific distribution can be applied if it is known. The range of the domain can be obtain form the remaining samples.
  - Expectation Maximization (EM) [2] - a sophisticated algorithm with many implementations.
  - k-nn — the unknown value is filled using k the most similar samples.
  - nn, fuzzy, etc. — the missing value is filled by a neural network or a fuzzy system where attributes with missing values are treated as the output attributes and the neural network or fuzzy system is learnt using other samples as the learning set.
  - multiple imputation — particular incomplete samples are replaced by a collection (cloud) of samples. Known values are copied and the missing values are many times randomly filled using assumed distribution [3].
- Hybrid methods, such as EM-SPCA [4], which fill missing places and reduce dimensionality.

Undoubtedly, the most important advantage of some forms of data preprocessing (marginalisation, imputation or hybrid methods etc.) is the freedom of choice of any classification method. However, this approach has also several disadvantages. As a result of deletion, some crucial information may be lost. Pair-wise deletion can lead to mentioned above situation where particular analysis based on different attributes lead to incoherent results. During imputation, a quality of the filled data is very important. This process may sometimes generate unreal samples. New, filled data is based on current samples, thus there is a risk of generating biased results [5]. Imputation methods require additional resources. The k-nn method has probably the largest memory demands, while EM-based algorithms may require a lot of computation time. The answer to mentioned problems are systems, that can directly work with missing values, such as the rough neuro-fuzzy classifiers (RNFC) [6], [7], [8].

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.232

The RNFCs are representatives of the second approach to process data with missing values, i.e. methods designed especially to work with missing values and general methods adapted to missing values. They can work without preprocessing when part of data is missing. Other examples of this approach are the C4.5 algorithm [9], the rough k-nn algorithm [10] and their ensembles [11], [12].

The RNFCs are specific neuro-fuzzy classifiers which, due to applying the rough set theory [13], are adapted to classify objects or states described by a vectors containing the interval values as well as the lacks. The neuro-fuzzy classifiers (generally the neuro-fuzzy systems) are the implementation of fuzzy classifiers in a network form. The first neuro-fuzzy system was proposed by Wang and Mendel [14]. They applied the conjunction type of reasoning, which is known now as Mamdani–type reasoning, and centre average (CA) defuzzification. Then, other type of such system was considered. Czogała and Łęski [15] studied neuro-fuzzy systems (also classifiers) using modified indexed centre of gravity (MICOG) defuzzification and various type of reasoning. Rutkowska and Nowicki [16] developed the family of such systems applying discrete centre of gravity (DCOG) defuzzification and also various types of reasoning. The rough neuro-fuzzy classifiers [6], [7], [8] are built as a couple of the neuro-fuzzy classifiers combined in a specific way. The considerations presented in the paper are focused on RNFC realising the CA defuzzification and Mamdani-type of reasoning, as in [6]. However, the presented results concern also other RNFCs, such as the ones presented in [7] realising MICOG defuzzification together with the logical reasoning, and [8] realising DCOG defuzzification with Mamdani and logical reasoning.

The incomplete data can be used in a knowledge extraction process in several ways. The simplest one is to apply any of the aforementioned imputation methods as a preprocessing of any rule generating or gradient learning algorithms. The more sophisticated algorithms, such as subsequent versions of LEM [17], [18], [19] and LBR [20], or specific gradient methods [21], [22] are able to process incomplete samples directly. They can by applied to various rule base systems. Some algorithms are dedicated to specific systems, e.g. to Bayesian networks [23].

In all previous papers referred to RNFCs, it has been assumed that the knowledge applied in the considered classifier comes from the correctly working neuro-fuzzy classifier which realises the same methods of fuzzification, reasoning and defuzzification. Learning of RNFC "from scratch" using the incomplete data has not been presented. The rough neuro-fuzzy system (RNFS) designed for regression modelling has been considered in [24] as well as a method of calculating rule elements in the case of missing values. Learning of RNFS was also considered in [25].

In this paper a gradient learning of the rough neuro-fuzzy classifier (RNFC) parameters using a set of samples with missing values is proposed. It is worth mentioning that gradient methods applied to rule based systems are generally criticised because they produce uninterpretable sets of rules. Despite that, they are widely used, mainly in the final phase of system design, but not only.

The main contribution of the paper is the mentioned above gradient learning adapted to the rough neuro-fuzzy classifiers. Moreover, the paper contains other original solutions related to the main topic. The former architecture of RNFCs has been extended with two new methods of the rough fuzzy answer reduction which gives final crisp decision (see eq. (9) and (10)). As the element of the new learning procedure, a method of error calculation (see (20)) was proposed. They relate to the methods of answer reduction. Some important aspects of an error propagation is disputed, e.g. propagation through the nondifferentiable elements, such as $\inf$ and $\sup$ (see eq. (7), (8) and (20)). The initialisation of RNFC parameters is realised based on Partial Data Strategy Fuzzy c-Means Clustering [26]. The method of RNFC evaluation has been also changed to be suited for other elements of the system. The proposed methods takes into account a problem of non-equinumerous classes in a training or validating sets. The specific organisation of the learning process has been proposed (Section III). The accuracy of learnt RNFC is compared with the results obtained for k–nn, which has been also adapted to the case of missing values. The overall learning and evaluating scheme is as follows

- Dataset preparation with constant or changing level of missing values,
- RNFC initialization by PDSFcMC,
- Novel gradient learning method for RNFC,
- Evaluation of RNFC with WTA method.

The paper is organised as follows. Section II presents the basic architecture of RNFC. It is used in further parts of the article. The details of the gradient learning designed for RNFCs are presented in the section III and Section IV contains some aspects and results of experiments. The contributions and conclusions are summarised in Section V.

II. Rough neuro-fuzzy classifier

Rough neuro-fuzzy classifiers are a solution to classification based on incomplete input information. They are a special form of neuro–fuzzy systems adapted to the rough set theory. There are many versions of such systems as they can use various methods of fuzzification, reasoning and defuzzification. The simplest one, described in [6], applies singleton fuzzification, Mamdani–type reasoning and centre average (CA) defuzzification method. The considered system classifies object x described by n attributes $\mathbf{v} = [v_1, \ldots, v_n]$ with values $\overline{\mathbf{v}} = [\overline{v}_1, \ldots, \overline{v}_n]$. Some of the values can be missing. The goal is to determine the membership of the object to m classes ω_j, $j = 1, \ldots, m$. Similarly to other fuzzy and neuro-fuzzy systems, the rough neuro-fuzzy classifiers use knowledge in the form of N fuzzy rules

$$\begin{array}{c} R^r: \textbf{IF } v_1 \text{ is } A_1^r \text{ AND } \ldots \text{AND } v_n \text{ is } A_n^r \\ \textbf{THEN } x \in \omega_1(\overline{z}_1^r), x \in \omega_2(\overline{z}_2^r) \ldots x \in \omega_m(\overline{z}_m^r), \end{array} \quad (1)$$

where $r = 1, \ldots, N$ is the rule number, A_i^r are the fuzzy sets used in antecedent of r–th rule and refers to input attribute v_i, $\overline{z}_j^r$ determine the membership of object x to class ω_j according to r–th rule. In the case of classification it is assumed that $\overline{z}_j^r \in \{0, 1\}$, i.e. $x \notin \omega_j$ or $x \in \omega_j$, when the conditions specified in the antecedent of r–th rule are met. When all,

provided in the classifier, input values are known the fuzzy output of the appropriate neuro–fuzzy classifier is given by

$$\mu_{\omega_j}(x) = \overline{z}_j = \frac{\sum_{r=1}^{N} \overline{z}_j^r \mu_{A^r}(\overline{\mathbf{v}})}{\sum_{r=1}^{N} \mu_{A^r}(\overline{\mathbf{v}})}, \tag{2}$$

where $A^r = A_1^r \times A_2^r \times \ldots \times A_n^r$ is a Cartesian product of fuzzy sets from the antecedent of r–th rule.

In the case of rough neuro-fuzzy classifier, in contrast to neuro-fuzzy and fuzzy classifiers, not all values $\overline{v}_i$ must be known. Thus, the set of all n input attributes, denoted by Q, is divided into two subsets — the set of attributes with known values and the set of attributes with missing values — denoted by P and G, respectively. Obviously, $P \cup G = Q$ and $P \cap G = \emptyset$. The answer of the rough neuro-fuzzy classifier takes the form of m intervals $\left[\underline{\overline{z}_j}, \overline{\overline{z}_j}\right]$, determined for all sets ω_j. The beginning and the end of the interval are specified as follows

$$\underline{\overline{z}_j} = \frac{\sum_{r=1}^{N} \overline{z}_j^r \mu_{\underline{\widetilde{P}}A^r}(\overline{\mathbf{v}})}{\sum_{r=1}^{N} \left(\overline{z}_j^r \mu_{\underline{\widetilde{P}}A^r}(\overline{\mathbf{v}}) + \neg \overline{z}_j^r \mu_{\overline{\widetilde{P}}A^r}(\overline{\mathbf{v}}) \right)} \tag{3}$$

and

$$\overline{\overline{z}_j} = \frac{\sum_{r=1}^{N} \overline{z}_j^r \mu_{\overline{\widetilde{P}}A^r}(\overline{\mathbf{v}})}{\sum_{r=1}^{N} \left(\overline{z}_j^r \mu_{\overline{\widetilde{P}}A^r}(\overline{\mathbf{v}}) + \neg \overline{z}_j^r \mu_{\underline{\widetilde{P}}A^r}(\overline{\mathbf{v}}) \right)}, \tag{4}$$

according to [6]. The negation operator $\neg$ is defined as $\neg \overline{z}_j^r = 1 - \overline{z}_j^r$. A crucial role in calculating the values $\underline{\overline{z}_j}$ and $\overline{\overline{z}_j}$ play the $\widetilde{P}$–lower and $\widetilde{P}$–upper approximations of antecedent fuzzy sets A_i^r denoted as $\underline{\widetilde{P}}A^r$ and $\overline{\widetilde{P}}A^r$, respectively. They designate the rough fuzzy sets [27] which approximate sets A_i^r. The corresponding membership functions are defined following the Dubois and Prade [28], i.e.

$$\mu_{\underline{\widetilde{P}}A^r}(\overline{\mathbf{v}}) = \underset{\overline{\mathbf{v}}_G \in \mathbb{V}_G}{inf} \mu_A^r(\overline{\mathbf{v}}_P, \overline{\mathbf{v}}_G) \tag{5}$$

and

$$\mu_{\overline{\widetilde{P}}A^r}(\overline{\mathbf{v}}) = \underset{\overline{\mathbf{v}}_G \in \mathbb{V}_G}{sup} \mu_A^r(\overline{\mathbf{v}}_P, \overline{\mathbf{v}}_G), \tag{6}$$

where $\overline{\mathbf{v}}_P$ and $\overline{\mathbf{v}}_G$ are the subvectors containing known and missing values, respectively. $\mathbb{V}_G$ is the domain of unknown input values. Functions (5) and (6) can be also defined using any t–norm in definition of Cartesian product [7]

$$\mu_{\underline{\widetilde{P}}A^r}(\overline{\mathbf{v}}) = T\left(\underset{i:v_i \in P}{T}(\mu_{A_i^r}(\overline{v}_i)), \underset{i:v_i \in G}{T} \inf_{\overline{v}_i \in \mathbb{V}_i} \mu_{A_i^r}(\overline{v}_i) \right) \tag{7}$$

and

$$\mu_{\overline{\widetilde{P}}A^r}(\overline{\mathbf{v}}) = T\left(\underset{i:v_i \in P}{T}(\mu_{A_i^r}(\overline{v}_i)), \underset{i:v_i \in G}{T} \sup_{\overline{v}_i \in \mathbb{V}_i} \mu_{A_i^r}(\overline{v}_i) \right), \tag{8}$$

where $\mathbb{V}_i$ is the domains of feature v_i.

When values of all input attributes are known ($P = Q$ and $G = \emptyset$), the rough neuro-fuzzy classifier is reduced to the suitable neuro-fuzzy classifier (see eq. (2)) and the output interval is reduced to single values, i.e. $\overline{z}_j = \underline{\overline{z}_j} = \overline{\overline{z}_j}$. When some input values are missing then $\underline{\overline{z}_j} \leq \overline{z}_j \leq \overline{\overline{z}_j}$ occurs, where $\overline{z}_j$ is the value corresponding to the case of all known values. When none value is known ($P = \emptyset$ and $G = Q$), the answer becomes intervals which contain all acceptable output values, i.e. $\left[\underline{\overline{z}_j}, \overline{\overline{z}_j}\right] = [0, 1]$. Fig. 1 presents the architecture of such rough neuro–fuzzy classifier. The antecedent part is common for all output classes, the consequent part should be duplicated for each class.

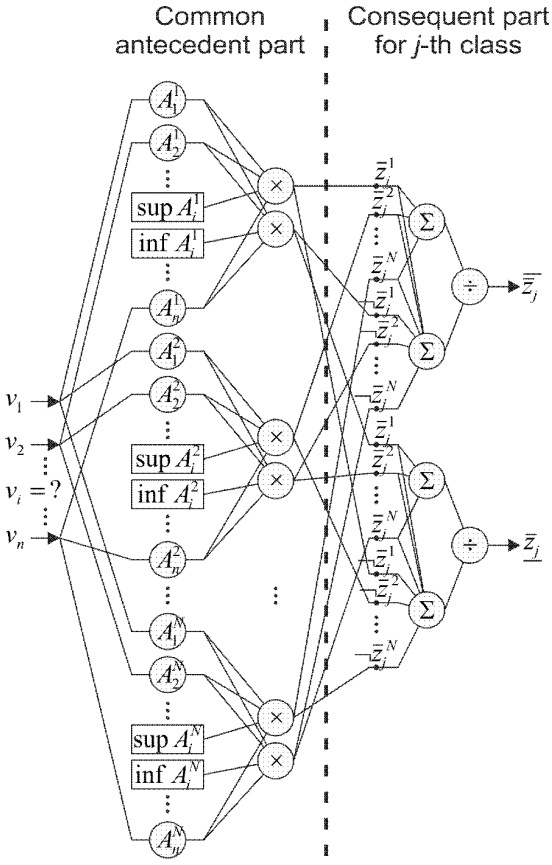

Fig. 1. Scheme of rough neuro–fuzzy architecture.

Finally, the fuzzy interval output obtained from rough neuro–fuzzy classifier should be reduced to a crisp decision. The methods of the reduction can be complex, but the simple one is following [7]

$$\begin{cases} x \in \omega_j & \text{if } \overline{z}_j \geq \frac{1}{2} \text{ and } \overline{\overline{z}_j} > \frac{1}{2} \\ x \notin \omega_j & \text{if } \underline{\overline{z}_j} < \frac{1}{2} \text{ and } \overline{\overline{z}_j} \leq \frac{1}{2} \\ \text{not known} & \text{in other cases.} \end{cases} \tag{9}$$

The last case ("not known") indicates that the input information is too poor and some missing values must be obtained and entered into the classifier. This case does not occur when all input values are known.

In the paper, new methods of reduction are applied. The first one is similar to the proposed in definition (9), but more restrictive. It is defined as follows

$$\begin{cases} x \in \omega_j & \text{if } \overline{z}_j \geq \frac{1}{2} \text{ and } \overline{\overline{z}_j} > \frac{1}{2} \text{ and} \\ & \forall j' \neq j, \overline{z}_{j'} < \frac{1}{2} \text{ and } \overline{\overline{z}_{j'}} \leq \frac{1}{2} \\ x \notin \omega_j & \text{if } \underline{\overline{z}_j} < \frac{1}{2} \text{ and } \overline{\overline{z}_j} \leq \frac{1}{2} \\ \text{not known} & \text{in other cases.} \end{cases} \tag{10}$$

It protects against the indication by the classifier more than one class. We use also a WTA method which is less restrictive. It is defined as follows

$$\begin{cases} x \in \omega_j & \text{if } \forall j' \neq j, \underline{\overline{z}_j} + \overline{\overline{z}_j} > \underline{\overline{z}_{j'}} + \overline{\overline{z}_{j'}} \\ x \notin \omega_j & \text{if } \exists j' \neq j, \underline{\overline{z}_j} + \overline{\overline{z}_j} < \underline{\overline{z}_{j'}} + \overline{\overline{z}_{j'}} \\ \text{not known} & \text{in other cases.} \end{cases} \tag{11}$$

The last case in definition (11) occurs only when the highest value of medium of interval is indicated for more than one set ω_j. The number of indicated sets can be limited to one by some restrictions in the rule base as well as using the reduction depicted in definition (9).

III. Gradient learning of RNFCs

In the case of feedforward neural networks the gradient learning is implemented as the backpropagation method. The network is trained mapping the input vectors $\overline{\mathbf{v}} = [\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_n]$ into the vectors of desired outputs $\mathbf{d} = [d_1, d_2, \ldots, d_m]$. Both are elements of samples which are contained in training (learning) set. As a result of assumed optimization criterion, usually

$$Q = \frac{1}{2} \sum_{j=1}^{m} (d_j - y_j)^2, \tag{12}$$

the errors

$$\epsilon_j = d_j - y_j \tag{13}$$

are calculated on the outputs of the last layer of the network and then they are propagated back to the weights, which are then corrected. The same method has been applied for the neuro-fuzzy systems [29]. The methodology is now commonly used when samples $\{\overline{\mathbf{v}}, \mathbf{d}\}$ are complete or incomplete, but preprocessed using any imputation or marginalisation, with all consequences of using such methods mentioned in Section I.

In the paper the gradient learning is adapted to train the rough neuro-fuzzy classifiers using the incomplete samples — the vector $\overline{\mathbf{v}}$ has constant length, but some of its elements could be missing. Due to consideration the classification case it is assumed that each learning sample belongs to exactly one class, so $d_j \in \{0, 1\}$ and $\sum_{j=1}^{m} d_j = 1$.

In the case of the rough neuro–fuzzy system, the imputation and marginalisation is not needed. However, the error on the output layer is redefined due to the interval nature of the system answer.

A. Error calculation

The answer of the rough neuro-fuzzy classifier has the form of intervals $\left[\underline{\overline{z}_j}, \overline{\overline{z}_j}\right]$ obtained for all output classes ω_j. Thus, the error can be also intervals, i.e. $\left[\underline{\epsilon}_j, \overline{\epsilon}_j\right]$. Below, a new way of calculating the error is proposed. In contrary to [25], only one optimization criterion is used. It is the general criterion (12), but taking into account both the lower and upper output value for each class, with changed definition of distance and desiring value d. Thus, the criterion is defined as follows

$$Q = \frac{1}{2} \sum_{j=1}^{m} \left(\rho^2 \left(\underline{\overline{z}_j}, \underline{D_j} \right) + \rho^2 \left(\overline{\overline{z}_j}, \overline{D_j} \right) \right) \tag{14}$$

where ρ is a distance between value z and non-empty set D

$$\rho(v, D) = inf_{d \in D} |v - d|. \tag{15}$$

The single value d can be treat as a singleton set $D = \{d\}$. We can define a desired interval with additional parameter $\mathfrak{p} \in [0, 1]$ allowing to adjust the width of the interval

$$\underline{D_j} = \begin{cases} \{0\} & \text{if } x \notin \omega_j \\ [1 - \mathfrak{p}, 1] & \text{if } x \in \omega_j, \end{cases} \tag{16}$$

$$\overline{D_j} = \begin{cases} [0, \mathfrak{p}] & \text{if } x \notin \omega_j \\ \{1\} & \text{if } x \in \omega_j. \end{cases} \tag{17}$$

We further extended definition of interval (16) by a heuristic element taking into account the width of output interval $[\underline{\overline{z}_j}, \overline{\overline{z}_j}]$. It is also a form of a feedback. Now the interval is defined

$$\underline{D_j} = \begin{cases} \{0\} & \text{if } x \notin \omega_j \\ \left[\max \left\{ 1 - \mathfrak{p}, 1 - \left(\overline{\overline{z}_j} - \underline{\overline{z}_j} \right) \right\}, 1 \right] & \text{if } x \in \omega_j \end{cases} \tag{18}$$

$$\overline{D_j} = \begin{cases} \left[0, \min \left\{ \mathfrak{p}, \overline{\overline{z}_j} - \underline{\overline{z}_j} \right\} \right] & \text{if } x \notin \omega_j \\ \{1\} & \text{if } x \in \omega_j. \end{cases} \tag{19}$$

Thus, the error is calculated as follows

$$\underline{\epsilon}_j = \begin{cases} 0 - \underline{\overline{z}_j} & \text{if } x \notin \omega_j \\ \max \left\{ 1 - \mathfrak{p}, 1 - \left(\overline{\overline{z}_j} - \underline{\overline{z}_j} \right) \right\} - \underline{\overline{z}_j} & \text{if } x \in \omega_j \end{cases} \tag{20}$$

$$\overline{\epsilon}_j = \begin{cases} \min \left\{ \mathfrak{p}, \overline{\overline{z}_j} - \underline{\overline{z}_j} \right\} - \overline{\overline{z}_j} & \text{if } x \notin \omega_j \\ 1 - \overline{\overline{z}_j} & \text{if } x \in \omega_j. \end{cases} \tag{21}$$

B. Learning procedure

There are three sets of parameters inside the architecture of the rough neuro-fuzzy classifier (Section II) which can be adjusted during learning. In the case of Gaussian antecedent fuzzy sets

$$\mu_{A_i^r}(\overline{v}_i) = \exp\left(- \left(\frac{\overline{v}_i - \hat{v}_i^r}{\sigma_i^r} \right)^2 \right), \tag{22}$$

the parameters are positions of their centres — $\hat{v}_i^r$, and their spreads — σ_i^r. Moreover, the consequents of rules are represented by parameters $\overline{z}_j^r$.

According to the gradient learning (the steepest descent method) principle, all the parameters are adjusted by following value [29]

$$\Delta w = -\eta \frac{\partial Q}{\partial w}, \tag{23}$$

where Q is the criterion defined in eq. (14), η is a learning coefficient (could be stable or unstable during the process), and w represents each of the mentioned above, parameters ($\hat{v}_i^r$, σ_i^r, and $\overline{z}_j^r$). From a formal point of view, the adjustment defined in eq. (23) cannot be derived. This is due to non-differentiable functions, i.e $\inf$, $\sup$, $\min$, and $\max$ present in the RNFC architecture and the error definitions (see e.g. eq. (7), (8), and (20)). For purpose of the error propagation, the specific substitutions were proposed. For example, for the $\inf$ function they are defined as follows

$$\frac{\partial \inf\limits_i v_i}{\partial v_k} := \begin{cases} 0 & \text{if } \inf\limits_i v_i \neq v_k \\ 1 & \text{if } \inf\limits_i v_i = v_k. \end{cases} \tag{24}$$

C. The initializing of rough neuro-fuzzy classifier

A common procedure is to initialize learning systems by random values. A family of c-means clustering algorithms is often used for this purpose. In the the paper the RNFC is initialized using the Partial Data Strategy Fuzzy c-Means Clustering (PDSFcMC) [26]. This method works separately for each class and derives parameters of the clusters in the class. Thus, computed clusters are converted into fuzzy sets linked with rules, with appropriate class pointed in the antecedent. In general, the rules (1) and the antecedent fuzzy sets can be arbitrarily complex. However, in the paper it is assumed that each rule corresponds to a single cluster. So, the number of clusters is equal to the number of rules in each class. The second consequence of such procedure is that the rules in the network indicate exactly one class. Before clustering procedure, dataset is normalised and then a reversal process is executed on the clusters.

D. Evaluation of RNFC efficiency

As the result of assumed methods of reducing the fuzzy output to crisp decision, i.e. (10) and (11), RNFC classifies the object x to single class. It is also assumed that the samples in learning and testing sets belongs to single class as well. The evaluation of the classifier is based on single samples classification correctness. The binary indicator g_c_s is proposed. It is set to 1 only if the sample x_s is classified to the proper class, i.e.

$$g\_c_s = \begin{cases} 1 & \text{if } \forall j = 1,..,m \begin{pmatrix} d_{j,s} = 1 \Rightarrow x_s \in \omega_j \\ \wedge \\ d_{j,s} = 0 \Rightarrow x_s \notin \omega_j \end{pmatrix} \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

where $d_{j,s}$ is the desired membership (0 or 1) of sample x_s to class ω_j while $x_s \in \omega_j$ and $x_s \notin \omega_j$ are the final decisions of the classifier under evaluation. An efficiency of the network is measured as a mean of accuracy of classification for each class,

$$\textit{efficiency} = \frac{1}{m} \sum_{j=1}^{m} \frac{1}{\parallel \omega_j \parallel} \sum_{s: x_s \in \omega_j} g\_c_s, \tag{26}$$

where $\parallel \omega_j \parallel$ is a number of samples in a class ω_j and g_c_s determines if sample x_s is correctly classified.

IV. Experiments and results

The proposed classifier designing method was tested using the Breast Cancer Wisconsin dataset from UCI repository [30], which has 0.25% missing values. Full 10-fold cross-validation was repeated five times for each experiment. Every time the simulated missing data were different. For the experiments purposes, single values were removed in a pseudo random style, taking into account the level of missing values. The algorithm enforced the same level of missing data in every class. Thanks to this procedure, fluctuation of results caused by various distribution of missing values is minimised and the MCAR case is simulated. A modified k-nn classifier was used as the reference system in evaluation of RNFC. It was examined in the same way as RNFC, i.e. 5 times in 10-fold cross-validation procedure. This experiment has been performed for several small values of neighbours number. In such range of parameter k, its value had rather small impact on the efficiency of classification. Despite its simplicity, this system showed quite good performance, even when the level of missing values was high (Fig. 2). However, the level of missing features has decidedly negative influence to classification efficiency. It concerns both learning and testing sets and have been confirmed in all tests.

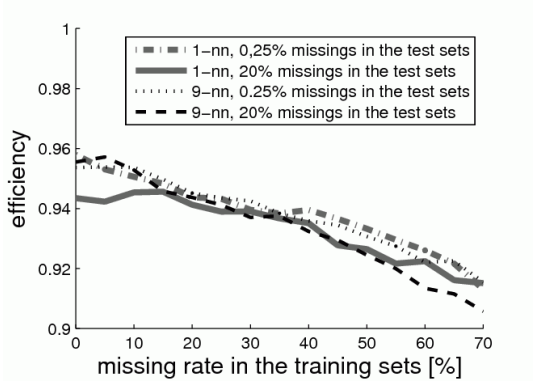

Fig. 2. Efficiency of k-nn algorithm for the Breast Cancer Wisconsin dataset versus missing values level in a training set.

RNFC was tested using WTA reduction method (11) and appropriate method of evaluation (25). The system was learnt default with constant parameter $\mathfrak{p}$, number of epochs, amount of rules per class, but the level of missing values was variable. In all the cases, the 10-fold CV procedure was repeated 5 times. RNFC was initialised by PDSFcMC [26] what yielded good initial system parameter values for further learning process. Experiments showed that in a wide range of missing feature level, even up to 45%, missing data used during system initialisation has little negative effect to the in the further work of the classifier. Moreover, we observed that the efficiency of classification of the data with missing features is improved when a classifier is prepared using data with missing features as well. It holds when level of missing features applied to built the classifier does not exceed some value. In the case of the applied data set, it is about 45%.

A. Learning RNFC with fixed level of missing values

This part of the experiments was performed with the fixed level of missing values, both in the learning and testing datasets. It allowed to examine the impact of the error computing method and parameter $\mathfrak{p}$ used in the proposed learning method (Section III). The level of missing values was fixed to 10%. Moreover, the learning coefficient η was empirically set to 0.1.

An influence of the learning parameter $\mathfrak{p}$ is shown in Fig. 3. The network was tested with 50% missing data in the learning set and 7.5% in the testing set.

B. Learning RNFC with changing level of missing values

The system was learned on datasets with various level of missing values, with 10-fold cross validation repeated five times for two rules per class. By default the WTA (11)

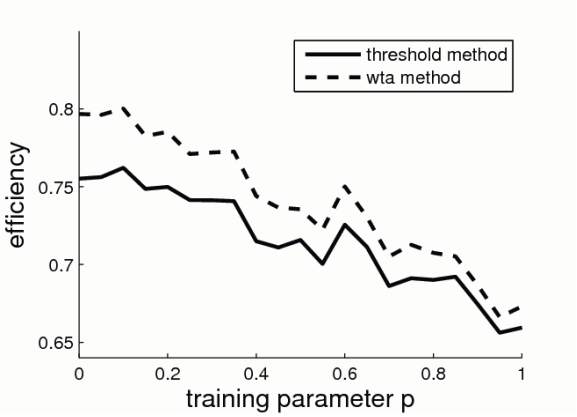

Fig. 3. Impact of p on the classification efficiency for the Breast Cancer Wisconsin dataset with 50% missing data in the training sets and 7.5% in the testing sets and two rules per class.

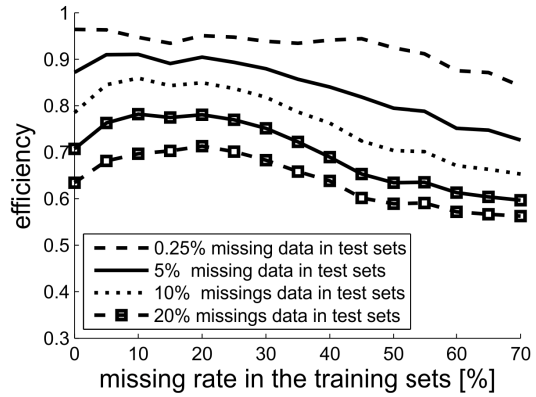

Fig. 4. Classification efficiency of RNFC for the Breast Cancer Wisconsin dataset and two rules per class.

reduction method is used as the efficiency reached using the WTA method was almost always better than using the method described by (10). The results are shown in Fig. 4. An influence of the number of rules and the level of missing values on the effectiveness is shown in Table I. The results show that increase in the number of rules per class above two has small impact on the efficiency. Thus, the next tests are prepared using 2 rules per class. The efficiency for a wide range of missing feature level is shown in Fig. 5.

The relationship between the level of missing values in the test set and reached efficiency is an interesting phenomenon. It is particularly evident when the level of missing vales in training set is significant. In most cases the relationship can be divided into following three parts (illustrated as "phases" in Fig. 6):

1) The efficiency increases with increasing level of missing values. This phase is more visible with a higher level of lacks in a test set and not visible when the level is marginal.
2) The efficiency fluctuates or decreases slowly.
3) The efficiency falls rapidly for high level of missing values.

TABLE I. EFFICIENCY OF RNFC ON THE BREAST CANCER WISCONSIN DATASET CLASSIFICATION

missing rate in test sets	missing rate in the training sets: 0.25%	5%	10%	20%
2 rules for every class				
0.25%	0.97	0.96	0.95	0.95
5%	0.94	0.95	0.94	0.93
10%	0.89	0.92	0.94	0.90
20%	0.78	0.85	0.89	0.84
3 rules for every class				
0.25%	0.97	0.97	0.96	0.96
5%	0.93	0.94	0.94	0.92
10%	0.89	0.91	0.91	0.89
20%	0.78	0.81	0.83	0.80
4 rules for every class				
0.25%	0.96	0.97	0.96	0.95
5%	0.92	0.94	0.94	0.94
10%	0.88	0.92	0.93	0.93
20%	0.77	0.85	0.87	0.88

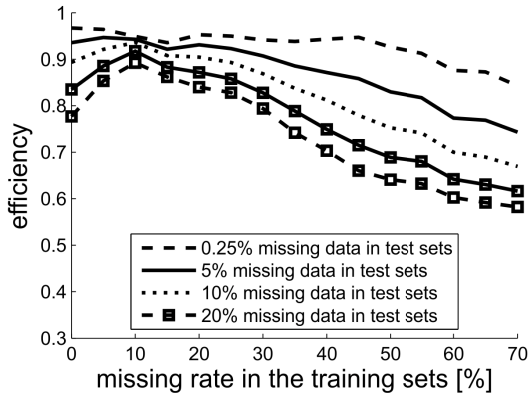

Fig. 5. Efficiency of RNFC after learning on the Breast Cancer Wisconsin dataset with two rules per class and various level of missing data.

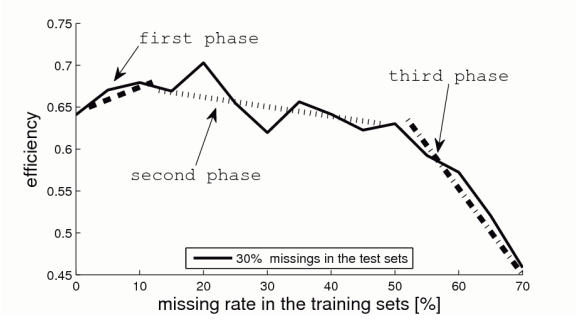

Fig. 6. Example of the relationship between the level of missing values in the test set and classification efficiency of RNFC.

V. Conclusion

The paper presents a complete procedure of data-driven rough-neuro-fuzzy classifier design. It starts from initialising the network structure and finishes on the gradient learning and testing.

The original element of this process is the gradient learning adapted to a specific architecture of the RNFC which works with incomplete learning data. The proposed method consists of several elements. The model of the rough-neuro-fuzzy decision system has been extended by two new methods of crisp decision determination. A specific error calculation method have been defined. In the proposed method of learning it is assumed that in the case of incomplete input data (missing values) the primary answer of the system is the interval with non-zero width. The single value (zero width interval) is desired only in the case of complete input data and it is forced by the architecture of the system. Such solution reflects the main feature of RNFC which manifests the lack of input information by the width of output interval. This is a unique feature of the proposed procedure.

The experiments which have been performed confirmed correctness of the concept and an expected behaviour of the classifier during the learning. Moreover, it shown the interesting properties of the learning process mentioned in Section IV. It could be valuable advice for decision system designers. Considerable influence on the learning process has also the applied method of treatment a non-equinumerous classes in learning and testing sets. It equalizes impact of classes represented by various number of samples on the global error value and the learning process.

The comparison between the proposed RNFC and the k–nn classifier adapted to process incomplete data confirms that the k–nn method is a strong algorithm and it is difficult to vie with it in relation to the classification efficiency level. Thus, the results obtained by the k–nn algorithm have been used to check the correctness of RNFC creating process. Of course, the k-nn method is much more computationally demanding.

The methodology proposed in the paper can be also applied together with other concepts of decision systems. The ensembles of classifiers [31], [32] seems to be particularly interesting. Such systems can be learnt using the backpropagation algorithm [33]. It can be expected that also other neuro–fuzzy systems, e.g. relational [32], [34], [35], flexible [36], [37] and type 2 [38], [39], can be adapted to work with incomplete data and can be learnt or tuned by the proposed methodology.

Acknowledgment

This work was supported by the Polish National Science Centre (NCN) within project number DEC-2011/01/D/ST6/06957.

References

[1] R. Little and D. Rubin, *Statistical Analysis with Missing Data*. John Wiley and Sons, 1987.

[2] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," in *Journal of the Royal Statistical Society. Series B*, vol. 39, 1977, pp. 1–38.

[3] N. Sartori, A. Salvan, and K. Thomaseth, "Multiple imputation of missing values in a cancer mortality analysis with estimated exposure dose," *Computational Statistics and Data Analysis*, vol. 49, no. 3, pp. 937–953, 2005.

[4] I. Stanimirova, M. Daszykowski, and B. Walczak, "Dealing with missing values and outliers in principal component analysis," *Talanta*, vol. 72, no. 1, pp. 172–178, 2007.

[5] D. A.R., van der Heijden G.J., S. T., and M. K.G., "Review: A gentle introduction to imputation of missing values," *Journal of Clinical Epidemiology*, vol. 59, pp. 1087–1091, 2006.

[6] R. Nowicki, "Rough–neuro–fuzzy structures for classification with missing data," *IEEE Trans. on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 39, no. 6, pp. 1334–1347, 2009.

[7] ——, "On combining neuro–fuzzy architectures with the rough set theory to solve classification problems with incomplete data," *IEEE Trans. on Knowledge and Data Engineering*, vol. 20, no. 9, pp. 1239–1253, Sep. 2008.

[8] ——, "On classification with missing data using rough-neuro-fuzzy systems," *International Journal of Applied Mathematics and Computer Science*, vol. 20, no. 1, pp. 55–67, 2010.

[9] J. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[10] R. K. Nowicki, B. A. Nowak, and M. Wozniak, "Rough k nearest neighbours for classification in the case of missing input data," in *Proceedings of the 9th International Conference on Knowledge, Information and Creativity Support Systems*, Limassol, Cyprus, Nov. 2014, pp. 196–207.

[11] M. Korytkowski, R. Nowicki, R. Scherer, and L. Rutkowski, "Ensemble of rough–neuro–fuzzy systems for classification with missing features," in *Proceedings of World Congress on Computational Intelligence 2008*, 2008, pp. 1745–1750.

[12] M. Korytkowski, R. Nowicki, L. Rutkowski, and R. Scherer, "Adaboost ensemble of dcog rough-neuro-fuzzy systems," *Lectures Notes in Computer Systems*, vol. 6922, pp. 62–71, 2011.

[13] Z. Pawlak, "Rough sets," *International Journal of Computer and Information Sciences*, vol. 11, no. 5, pp. 341–356, 1982.

[14] J. Wang, L.-X.; Mendel, "Back-propagation fuzzy system as nonlinear dynamic system identifiers," in *IEEE International Conference of Fuzzy System*, 1992, pp. 1409–1418.

[15] E. Czogała and J. Łęski, *Fuzzy and Neuro–Fuzzy Intelligent Systems*. Heidelberg, New York: Physica–Verlag, A Springer–Verlag Company, 2000.

[16] D. Rutkowska and R. Nowicki, "Implication-based neuro–fuzzy architectures," *International Journal of Applied Mathematics and Computer Science*, vol. 10, no. 4, pp. 675–701, 2000.

[17] J. W. Grzymala-Busse, "An overview of the LERS1 learning systems," in *Proceedings of the 2nd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, 1989, pp. 838–844.

[18] ——, "LERS — a system for learning from examples based on rough sets," in *Intelligent Decision Support: Handbook of Applications and Advences of the Rough Sets Theory*, R. Sowiski, Ed. Dordrecht: Kluwer, 1992, pp. 3–18.

[19] E. Orlowska, Ed., *Incomplete Information: Rough Set Analysis*, ser. Studies in Fuzziness and Soft Computing. Berlin Heidelberg: Springer-Verlag, 1998, vol. 13.

[20] L. Shu, M. Z. Wen, and H. Dan, "Methods of learning rules based on rough set: Lbr and lem3," in *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, vol. 2, july 2001, pp. 753 –756 vol.2.

[21] T.-P. Hong, L.-H. Tseng, and B.-C. Chien, "Learning fuzzy rules from incomplete quantitative data by rough sets," in *Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on*, vol. 2, 2002, pp. 1438 –1443.

[22] ——, "Learning coverage rules from incomplete data based on rough sets," in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 4, oct. 2004, pp. 3226 – 3231 vol.4.

[23] F. Tian, H. Zhang, and Y. Lu, "Learning bayesian networks from incomplete data based on emi method," in *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, nov. 2003, pp. 323 – 330.

[24] K. Simiński, "Neuroroughfuzzy approach for regression modelling from missing data," *Int. J. Appl. Math. Comput. Sci.,*, vol. 22, no. 22, pp. 461–476, 2012.

[25] R. Nowicki, B. Nowak, J. Starczewski, and K. Cpalka, "The learning of neuro-fuzzy approximator with fuzzy rough sets in case of missing features," in *Neural Networks (IJCNN), 2014 International Joint Conference on*, Jul. 2014, pp. 3759–3766.

[26] J. Dixon, "Pattern recognition with partly missing data," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-9, pp. 617–621, 1979.

[27] D. Dubois and H. Prade, "Rough fuzzy sets and fuzzy rough sets," *International Journal of General Systems*, vol. 17, no. 2–3, pp. 191–209, 1990.

[28] ——, "Putting rough sets and fuzzy sets together," in *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, R. Sowiski, Ed. Dordrecht: Kluwer, 1992, pp. 203–232.

[29] L. Rutkowski, *Computational Intelligence. Methods and Techniques*. Springer–Verlag, 2008.

[30] C. J. Mertz and P. M. Murphy, "UCI respository of machine learning databases," Available online: *http://www.ics.uci.edu/pub/machine-learning-databases*.

[31] M. Korytkowski, L. Rutkowski, and R. Scherer, "From ensemble of fuzzy classifiers to single fuzzy rule base classifier," *Lecture Notes in Artificial Intelligence*, vol. 5097, pp. 265–272, 2008. [Online]. Available: http://www.springerlink.com/content/p2813172x2w414t2/

[32] R. Scherer, "Boosting ensemble of relational neuro-fuzzy systems," *Lecture Notes in Artificial Intelligence*, vol. 4029, pp. 306–313, 2006.

[33] M. Korytkowski, R. Scherer, and L. Rutkowski, "On combining backpropagation with boosting," in *2006 International Joint Conference on Neural Networks, Vancouver, BC, Canada*, 2006, pp. 1274–1277.

[34] R. Scherer, "Neuro-fuzzy relational systems for nonlinear approximation and prediction," *Nonlinear Analysis*, vol. 71, pp. e1420–e1425, 2009. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0362546X09001898

[35] ——, "Neuro-fuzzy systems with relation matrix," *Lecture Notes in Artificial Intelligence*, vol. 6113, pp. 210–216, 2010. [Online]. Available: http://springerlink.com/content/px5gux345l5l2xg4/

[36] L. Rutkowski and K. Cpałka, "A general approach to neuro - fuzzy systems," in *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*, vol. 3, Dec. 2001, pp. 1428–1431, december 2-5.

[37] ——, "A neuro-fuzzy controller with a compromise fuzzy reasoning," *Control and Cybernetics*, vol. 31, no. 2, pp. 297–308, 2002.

[38] J. Starczewski and L. Rutkowski, "Interval type 2 neuro-fuzzy systems based on interval consequents," in *Neural Networks and Soft Computing*, L. Rutkowski and J. Kacprzyk, Eds. Heidelberg, New York: Physica-Verlag, Springer-Verlag Company, 2003, pp. 570–577.

[39] L. Starczewski and L. Rutkowski, "Connectionist structures of type 2 fuzzy inference systems," *Lecture Notes in Computer Science*, vol. 2328, pp. 634–642, 2006.