Bogdan Gliwa*, Aleksander Byrski*

# HYBRID NEURO-FUZZY CLASSIFIER BASED ON NEFCLASS MODEL

*The paper presents hybrid neuro-fuzzy classifier, based on NEFCLASS model, which was modified. The presented classifier was compared to popular classifiers – neural networks and k-nearest neighbours. Efficiency of modifications in classifier was compared with methods used in original model NEFCLASS (learning methods). Accuracy of classifier was tested using 3 datasets from UCI Machine Learning Repository: iris, wine and breast cancer wisconsin. Moreover, influence of ensemble classification methods on classification accuracy was presented.*

**Keywords:** *Neuro-fuzzy classifier, NEFCLASS, neural networks, fuzzy systems*

# HYBRYDOWY NEURONOWO-ROZMYTY KLASYFIKATOR OPARTY NA MODELU NEFCLASS

*Artykuł przedstawia zasadę działania oraz wyniki badań eksperymentalnych klasyfikatora opartego na hybrydzie sieci neuronowej z logiką rozmytą, bazujący na modelu NEFCLASS. Prezentacja struktury i działania klasyfikatora została zilustrowana wynikami eksperymentów porównawczych przeprowadzonych dla popularnych klasyfikatorów, takich jak perceptron wielowarstwowy k najbliższych sąsiadów. Skuteczność wprowadzonych modyfikacji do klasyfikatora została porównana z metodami używanymi w oryginalnym modelu NEFCLASS (metody uczenia). Jako dane benchmarkowe posłużyły wybrane bazy danych z UCI Machine Learning Repository (*iris, wine, breast cancer wisconsin*). Zaprezentowano również wpływ użycia metod klasyfikacji zbiorczej na efektywność klasyfikacji.*

**Słowa kluczowe:** *klasyfikatory neuronowo-rozmyte, NEFCLASS, sieci neuronowe, systemy rozmyte*

## 1. Motivation

Algorithms and systems supporting automatic and semi-automatic classification has always been important and belonged to widely explored areas of computer science, and more precisely, machine learning. Flexibility of classification model, adaptability to different problems and easy parametrization are features that are always needed

* AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, IT and Electronics, Department of Computer Science, al. A. Mickiewicza 30, 30-059 Krakow, Poland, bgliwa@gmail.com, olekb@agh.edu.pl

for this kind of algorithms. However in some applications, such as medical decision support system, the possibility of explaining the reasons for the advices given by classifier becomes very important. Because of that, one should turn to the rule-based knowledge representation [14], as one of the most human-readable formats.

Neural networks are interesting mathematical models capable of being taught of different tasks such as classification, approximation, prediction and others [9]. Although they may be easily trained and based on already performed research some of their architectures may pose as universal computation models (see universal approximation theory for multi-layered perceptrons [9]), the representation of the knowledge in this area of artificial intelligence is not easily perceivable for human.

The fuzzy systems introduce the means of uncertainty to classification systems and may be used to effective processing the incomplete or uncertain data [25]. In addition, data representation in fuzzy systems may usually be very easily presented with use of rules.

In the course of this paper we would like to present the classification system based on NEFCLASS [22] model, that hybridizes the universal computation features of neural networks with uncertainty capabilities of data (described with use of rules) processing. In our opinion this approach enhances the possibility of interpretation of classification reasons, that is very important e.g. in the above-mentioned decision support systems (e.g. medical). After giving the short state-of-the-art and describing the structure and work of the classifier, we compare it with several, arbitrarily chosen classical classification algorithms, using popular benchmark datasets.

## 2. State of the art

In order to fit correctly our classifier into state of the art we would like to refer to several popular solutions.

Neural networks [27, 15] are simplified model of biological neural system. Most commonly used are MLP networks (multilayered perceptron) – feed-forward networks with powerful approximation possibilities (cf. universal aproximation theory [9]). The generalization possibilities makes them a good example of universal classifiers, taught under (and without) supervision, however one must consider dangers of overfitting or mislearning, depending on the parametrization.

*k-nearest neighbours classifier* [12] uses a concept of neighbourhood, defined in terms of distance vector to $k$ nearest vectors from learning dataset to the examined vector. Euclidean metric is most commonly used. The value of $k$ depends on dataset and usually must be chosen experimentally. Too low value of $k$ [28] makes the classifier sensitive to noise in data, too high value causes the neighbourhood to sprawl among objects from different classes.

SVM (Support Vector Machines) [3, 2] solves a binary classification task, but can be used to handle multi-class classification problem (one-against-all or one-against-one method). SVM finds optimal decision hyperplane that splits instances on 2 parts

– that maximize the margin between 2 classes. The classifier can transform original input space into higher dimensional space by nonlinear mapping. In transformed space there can be possible to construct linear decision surface. Such mapping function is called *kernel function*.

Decision trees [6] present process of making decision in the form of tree. In the nodes of decision tree, the clauses are placed, checking values of attributes. The categories may be found in the leafs. Conjunction of several conditions are represented in natural way in tree but alternatives make tree more complex. Complex decision trees are often simplified by using pruning. Structure of decision tree can be easily transformed into rule-based knowledge representation.

The above-mentioned classifier methods may be coupled together using ensemble classifiers, such as bagging and boosting, to achieve better performance.

In bagging [4, 13, 1] each weak classifier is trained on random redistribution of original dataset called *bootstrap sample*. Each bootstrap sample is generated by random sampling with replacement. Many vectors from original dataset may be repeated in bootsrap sample but others may be left out – approx. $\frac{1}{e} \approx 37\%$ of all samples is not presented in bootstrap sample. Response of such complex classifier is mostly determined by majority voting.

In boosting the algorithm [8, 24] gradually focuses on examples that are harder to classify. Initially weights of all instances have the same value. In the subsequent iterations the weights of misclassified samples are increased and weights of correctly classified samples are decreased. This method is called *reweighting*. But not all classifiers are adapted to decrease error on weighted dataset. Therefore, the alternative method is using – *resampling* [8]. Weights are still assigned to samples of learning dataset but on the basis of these weights the subsample is generated and the classifier operates on the subsample (not original dataset used for training). Selection of instances for such subsample is done according to the rule that samples harder to classify get greater probability of including them to that subsample. Each weak classifier has assigned *confidence index*: $\alpha = log(\frac{1-err}{err})$, where $err$ is sum of the weighted errors of all training samples. The confidence index is equivalent to weight of weak classifier.

To deal with uncertain or incomplete data, fuzzy sets [23, 17] may be used. They may be perceived as extension of classical sets, characterized by membership functions which express degree of object membership to a certain fuzzy set. For boundary values (0 and 1) of membership function the fuzzy set becomes classical, crisp set. Fuzzy approaches try to imitate human-like reasoning.

Achieving good classification accuracy is not sufficient in some areas – especially where the decisions made are irreversible (cf. medical diagnosis support). In such cases the classifier need to show human-readable explanations of decisions. It can be achieved by using proper knowledge representation, especially rule-based one seems well suited [14].

Of course comparing our classifier to all possible is too complex to be carried out efficiently and presented in one paper. That is why we chose several classical solutions and present the comparison in the last part of this contribution. Detailed testing should be carried out in context with a chosen problem, as it is impossible to prepare one solution to solve all possible problem with the same accuracy (cf. no free lunch theorem [5]).

## 3. NEFCLASS model

The NEFCLASS, neuro-fuzzy classifier was presented in [22]. NEFCLASS consists of three layers (input, rule and output) and can be treated as a specific neural network (see Fig. 1) – hence, elements of the model, are treated as neurons: input neurons, rule neurons and output neurons. We have chosen this model because of its synergy capabilities: rule-based representation and generalization possibilities of neural networks [18].

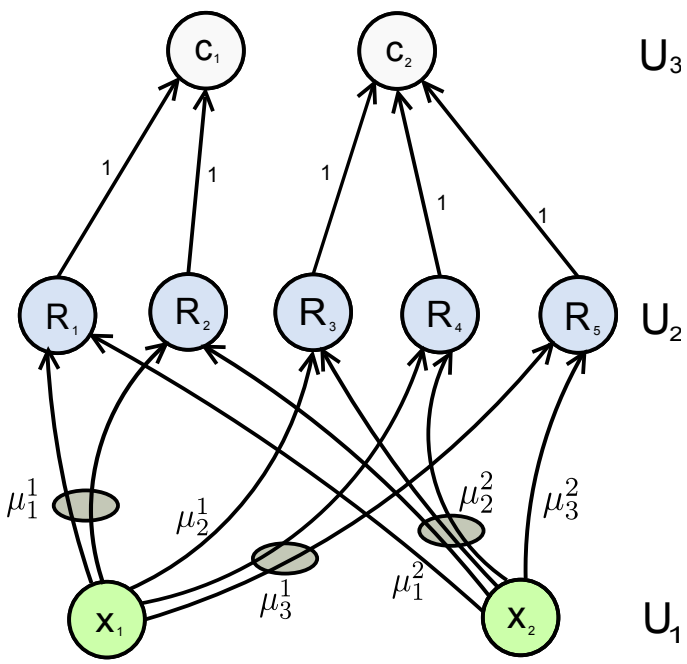


**Fig. 1.** Schema of NEFCLASS model

The layer $U_1$ processes input data. The activation function of neurons contained in $U_1$ is usually an identity function, but can also be different, e.g. in case of preprocessing with normalization [21].

The neurons of the hidden layer $U_2$ represent fuzzy rules. The NEFCLASS model is defined as neuro-fuzzy system of MAMDANI type [16], so rule form contains fuzzy

sets in its premises and conclusion. In the premises the fuzzy sets with triangular membership function are usually used, however, the conclusion of a rule is fuzzy set with singleton membership function (this is characteristic function of set that represents 1 as a fuzzy set). The premises of fuzzy rule become weights for the rule neurons of layer $U_2$. The conclusion of a rule is a connection from rule neuron to next layer.

As a activation function of rule neuron (to calculate of activation of rules on the basis of membership functions of premises) the T-norm is used (often this is minimum function):

$$a_R{}^p = \min_{x \in U_1} \{W(x, R)(a_x{}^p)\} \tag{1}$$

where $W(x, R)$ stands for the weight of the connection between input neuron $x$ and rule neuron $R$.

The weights for rule neurons (marked as $W(x, R)$) are shared – for each linguistic value the only one fuzzy set is used, assuring the feature of interpretability of the model. From every neuron of second layer, only one connection is attached to the third layer. This serves as connection between rule and class, being its conclusion. The connection has weight equal to 1 (marked as $W(R, c)$), because of the fact that fuzzy rules with weights are difficult to interpretation [20]. The autors also showed that modifications of rule weights can be replaced by modification of membership function and the use of weights is not necessary.

The layer $U_3$ is the output layer, activation function of output neurons (calculating activation value of given class on the basis of activations of rules that indicate given class as its conclusion) is T-conorm, usually maximum function:

$$a_c{}^p = \max_{R \in U_2} \{a_R{}^p\} \tag{2}$$

where $W(R, c) = 1$ is the weight of connection between rule neuron $R$ and output $c$.

After calculation of activation in output neurons, the neuron with highest activation is chosen as result of classification (cf. "winner takes all" method for unsupervised learning [9]).

## 3.1. NEFCLASS learning

Learning of the NEFCLASS model consists of 2 phases [18]:

- structural learning – creation of the rule set,
- iterative fuzzy sets learning – fuzzy sets optimization.

### 3.1.1. Algorithm of structural learning of NEFCLASS

Goal of this algorithm is to create rule set, that can be later tuned by fuzzy set learning algorithm. Pseudocode of the algorithm is presented in Alg. 1.

The process of the structural learning consists of three runs. In the first run the candidates set is created that consists of rules from learning data.

**Algorithm 1** NEFCLASS – the structural learning

**Require:** Z (learning dataset), $k_{max}$ (max number of rules)

1: $k \leftarrow 0$
2: **for all** $(p, t)$ in $Z$ **do**
3: For every input $x_i \in U_1$ find that membership function $\mu_{ji}$, that:

$$\mu_{ji}(p_i) = \max_{j \in \{1,..,q_i\}} \{{\mu_j}^i(p_i)\}$$

4: **if** $k < k_{max}$ and not exist rule $R$: $W(x_1, R) = {\mu_{j_1}}^1, ..., W(x_n, R) = {\mu_{j_n}}^n$ **then**
5: Create that rule and connect it to output according to $t$
6: Append created rule to candidate rule set
7: $k \leftarrow k + 1$
8: **end if**
9: **end for**
10: **for all** $(p, t)$ in $Z$ **do**
11: Calculate output value of NEFCLASS
12: For every rule calculate its accumulative activation value for class $t$
13: ▷ mark ${accumulative_j}^c$
14: **end for**
15: **for all** rule R **do**
16: **if** $\arg\max_c {accumulative_j}^c \neq consequent_j$ **then** ▷ consequent of rule $R$
17: $consequent_j \leftarrow \arg\max_c {accumulative_j}^c$
18: **end if**
19: $performance_R \leftarrow 0$
20: **end for**
21: **for all** $(p, t)$ in $Z$ **do**
22: Calculate output value of NEFCLASS
23: **for all** rule candidate **do**
24: $performance_R = performance_R + {a_R}^p * e_p$, where

$$e_p = \begin{cases} 1, & \text{if p is classified correctly} \\ -1, & \text{otherwise} \end{cases}$$

25: **end for**
26: **end for**
27: Choose rule set using one of the following procedures of rules selection:
„simple rules selection”
„best rules selection”
„best per class rules selection”

In the second run the conclusions of every rule are checked for correctness – if a rule has greater value of a sum of activations for other class than it has in its conclusion, then the conclusion changes to that class. In the third run the "performance" value is calculated for every rule:

$$performance_R = \sum_{p \in Z} {a_R}^p * e_p \tag{3}$$

where $e_p = \begin{cases} 1, & \text{if p is classified as correct,} \\ -1, & \text{otherwise.} \end{cases}$

Selection of rules is performed in three ways (the goal is to choose max $k_{max}$ rules) [18]:

- Simple rules selection (Alg. 2) – the first $k_{max}$ rules is chosen from the candidate rule set. This strategy can be successful if patterns has been chosen randomly from learning dataset and quantities of class instances in dataset are approximately equal.
- Best rules selection (Alg. 3) – the variant is proper if there are classes that should be represented by greater number of rules than others.
- Best per class rules selection (Alg. 4) – the variant is selected when the patterns are located in the same number of clusters for each class.

**Algorithm 2** NEFCLASS – simple rules selection

**Require:** $k_{max}$ (max number of rules), $j$ (number of rules in candidate rule set)
1: $k \leftarrow 0$
2: **while** $k < k_{max}$ **do**
3: $R = \arg\min_{R_j}\{j\}$
4: Add R to rule set
5: Remove R from candidate rule set
6: $k \leftarrow k + 1$
7: **end while**

**Algorithm 3** NEFCLASS – best rules selection

**Require:** $k_{max}$ (max number of rules), $j$ (number of rules in candidate rule set)
1: $k \leftarrow 0$
2: **while** $k < k_{max}$ **do**
3: $R = \arg\max_{R_j}\{performance_j\}$
4: Add R to rule set
5: Remove R from candidate rule set
6: $k \leftarrow k + 1$
7: **end while**

**Algorithm 4** NEFCLASS – best per class rules selection

**Require:** $k_{max}$ (max number of rules), $j$ (number of rules in candidate rule set), $m$ (number of classes)

1: $kPerClass \leftarrow \frac{k_{\max}}{m}$
2: **for all** class $c$ **do**
3: $k \leftarrow 0$
4: **while** $k < kPerClass$ **do**
5: $R = \underset{R_j, consequent_j = c}{\arg\max} \{performance_j\}$ ▷ consequent of rule $R$
6: Add R to rule set
7: Remove R from candidate rule set
8: $k \leftarrow k + 1$
9: **end while**
10: **end for**

### 3.1.2. Algorithm of fuzzy sets learning

Fuzzy set learning algorithm is an iterative procedure (Alg. 5), which is similar to backpropagation algorithm used to train multi-layered neural networks [9]. Gradient-based algorithm cannot be used here, because the membership fuzzy functions are not differentiable (triangle membership functions are used). During execution of the algorithm, the membership function is shifted and its support is increased or decreased. Very often the restrictions on these operations are defined, e.g. fuzzy sets must not pass each other or must not intersect more than 50% value of their support.

## 3.2. Pruning of rules

In order to increase readability of rules and decrease complexity of model the process of rules pruning is conducted. The following methods of rule pruning are usually performed in NEFCLASS model [22]:

- Pruning by correlation – attribute that has least influence on result is removed. To identify that attribute, the correlation or information gain is used.
- Pruning by classification frequency – rule that has largest degree of fulfillment in the smaller number of instances is removed. The classification accuracy is not worse if these cases are covered by other rules.
- Pruning by redundancy – linguistic variable, that has the smallest fulfillment degree in active rule in the smallest number number of cases, is removed. This pruning variant assumes that minimum operator is used for evaluation of rule premises. Hence, variable that always gives large fulfillment degree of premises does not affect activation value of the rule. In case of using other T-norm (instead of *min* operation) this strategy still can be used but it is less effective.
- Pruning by fuzziness – fuzzy set with largest support is found and all variables that uses it are removed from premises of every rule.

Automatic pruning is done by processing consecutively given variants of pruning. Modifications made by pruning variants are preserved only if they improve rule

**Algorithm 5** NEFCLASS – Fuzzy set learning

**Require:** Z (learning dataset), $\sigma$ (learning rate) $> 0$

**for all** $(p, t)$ in $Z$ **do**
  Calculate output value of NEFCLASS for vector $p$
  **for all** $c_i$ in $U_3$ **do**
    $\delta_{c_i} = t_i - a_{c_i}$
  **end for**
  **for all** rule $R$ in $U_2$ **do**
    **if** $a_R > 0$ **then**
      $\delta_R = a_R(1 - a_R) \sum_{c \in U_3} W(R, c)\delta_c$
      Find $x'$ that $W(x', R)(a_{x'}) = \min_{x \in U_1} \{W(x, R)(a_x)\}$
      $\mu = W(x', R)$ ▷ $a, b, c$ are parameters of $\mu$
      $\delta_b = \sigma\delta_R(c - a)\mathrm{sgn}(a_{x'} - b)$
      $\delta_a = -\sigma\delta_R(c - a) + \delta_b$
      $\delta_c = \sigma\delta_R(c - a) + \delta_b$
      Modify $\mu$ by values $\delta_a, \delta_b, \delta_c$ if they not violate given restrictions
    **end if**
  **end for**
**end for**

set (under different conditions). If modifications are not satisfactory, the rule set is restored to previous state.

Improving rule set is often understood as enhancing of classification accuracy (e.g. error decrease), reducing the model complexity (e.g. decreasing number of attributes) or combination of these two approaches. Usually this is an effect of a compromise between accuracy and simplicity of model. Large number of parameters is often needed to gain high accuracy, but it causes the model to become less understandable. However, reducing number of parameters sometimes enhances classification accuracy, because the model with large number of parameters can overfit the learning data and loose the generalization feature.

Side effects of the rule pruning can be inconsistencies in rule set that should be solved – it is done by removing invalid rules. According to [22], the rule set is consistent if does not consist:

- Contradictions – rules have different conclusions and their premises are the same or premises of one rule generalize premises of second one.
- Redundancies – rules with the same conclusions and the same premises or premises of first rule generalize premises of second one.

To speed up the process of fuzzy set learning, it should be conducted after pruning of rules and not after every step of pruning (if pruning process consists of several steps, and in every step different variant of pruning is executed). Sometimes process

of pruning is repeated several times before next iteration of fuzzy set learning is started – this method is known as exhaustive pruning.

### 3.3. Evaluation of model

First version of this model was proposed by Nauck and Kruse in 1995 [19]. In the original version the initial values of fuzzy sets parameters were needed to set by researcher and next were evaluated by iterative procedure similar to backpropagation training method. Later, there were approaches to automate this manual process. One of such try was using fuzzy clustering and based on this constructing fuzzy sets by projecting the clusters [10]. Next modification made by authors was handling of missing values and symbolic variables in dataset [22]. Another attempt to automate initial setting of fuzzy sets was using partitioning method in each dimension of vectors space based on entropy [11].

## 4. Hybrid neuro-fuzzy system

In the presented neuro-fuzzy system, fuzzy $c$-means clustering algorithm was used to extract initial fuzzy sets. In the consecutive steps, the model was trained according to above described NEFCLASS training with the modifications described below.

To prune rules the "pruning by classification frequency" method is used. Furthermore, there were proposed and implemented several new rule pruning and structure learning methods:

- Reduction of attributes – certain attribute is removed and the classification accuracy on the learning dataset is tested – if accuracy is not worse than defined value of percentage point, then this operation is approved, otherwise attribute is restored.
- Removing rules that reached maximum activation value, smaller than certain, defined value (the value's range is $[0, 1]$, usually the value 0.5 is used).
- Removing rules that differ in value of only one attribute and have the same conclusion. After removing one of them and the attribute that differs in second rule, the classification accuracy test on the learning dataset is performed – the acceptance condition is the same as in the first described modification – improving accuracy or slightly decreasing (but the number of rules is smaller, what makes this model more readable to human).
- New method of structural model learning is used, where the number of generated rules is proportional to the number of instances per class. In that way the classes with large number of instances have appropriately more rules than classes with small number of them. This method can be useful and better fits to data, in contrast to methods used in NEFCLASS model, especially in situations, where classes with more number of instances are concentrated around several clusters, whilst the classes with small number – around one of them.

- Second new method of learning (the most efficient one after conducting the experiments), where the maximum number of rules in the process of choosing rules to model is not defined – action are performed like in the simple method of structural learning. Afterward, the rules are evaluated, like in the classic NEFCLASS model i.e. during learning of fuzzy sets and the rule pruning. After passing several iterations the number of rules becomes acceptable. If not, the method of choosing $k$ final rules may be used.

  After pruning of rules, the following rules are removed:

- the rules that were not used even once on the learning dataset,
- redundant rules (these rules can exist after removing one attribute from every rule),
- rules that are specializations of other rules (the rule is treated as specializing other if the set of her premises is included in the promises set of the other and conclusions of both rules are the same).

If any rule is matches given pattern then the default class can be used (the class that has the highest number of instances from the learning dataset). The implemented system takes also advantage of boosting (SAMME boosting [29] has been used) and bagging [13] algorithm.

In order to test the classifier, a software framework was implemented. The framework can be configured using XML file, where the user can define the components and references among them. Therefore, the components may be tuned without need of code recompilation. Based on XML file content, appropriate dataset is loaded and tests are conducted. The constructed system is extensible and other classifiers can be included – currently besides neuro-fuzzy classifier, there are neural networks and $k$ nearest neighbours classifier. Two "ensemble classifiers" – *SAMME boosting* and *bagging*, were also implemented. The system was written in Java language. To implement environment, the Spring Framework[1] has been used. One of its well-known features is *Inversion of Control* – application container takes care of correct initialization of components and proper injection dependencies to them [7].

## 5. Experimental results

A comparison of neuro-fuzzy classifier with original NEFCLASS, neural network (multi-layered perceptron) and $k$ nearest neighbours has been conducted. After that, effects of changes of its certain parameters on classification accuracy were explored. One of such parameters is learning type and NEFCLASS model with original learning methods was compared with the new proposed ones. Although we consider only popular benchmark problems, we are aware, that detailed comparison of the proposed classifier should be evaluated on some real-world problem.

[1] http://www.springsource.org/about

## 5.1. Methodology of conducting tests

Experiments consisted of 10 runs of the system for every dataset and the results show average with standard deviation. The following datasets have been used for testing (datasets coming from *UCI Machine Learning Repository*[2]):

- *iris* – 4 attributes, 3 classes, 150 instances;
- *wine* – 13 attributes, 3 classes, 178 instances;
- *breast cancer wisconsin (original)* – 9 attributes, 2 classes, 683 instances.

Every dataset has been split randomly (during system running) into 2 following parts:

- learning part – 75% of all instances,
- testing part – 25% of all instances.

## 5.2. Classification results

Ensemble classifiers, that have been used in tests, consists of 3 NEFCLASS classifiers. In every iteration the rule pruning has been conducted (before and after learning of fuzzy sets). Acceptable difference in accuracy during process of rule pruning was equal to 1 – therefore, if accuracy after pruning rules was not worse than 1%, then operation was accepted. The new method without defining fixed number of rules has been used as structural learning method.

We used also multilayered perceptrons neural networks with trained using classical backpropagation algorithm. MLPs consisted of one hidden (contained 5 neurons), input and and output layers. Learning rate was constant in every iteration and its value was equal to 0.7, momentum value was 0.2 (also constant in every iteration). During conducted tests, neural networks performed 700 iteration in every run.

In $k$ nearest neighbour classifier as $k$ parameter the value 3 was established, so three nearest neighbours are taken into account during classification of given pattern. Every neighbour has the same influence on final decision of classifier – weights of neighbours are equal and have value 1 (majority voting).

Average times, given in tables, are average times of single run (average value of 10 runs of given classifier). In case of describing rules, character "?" means, that the marked attribute has no influence in process of classification (it can have any value). Depending on the number of fuzzy sets defined for given attribute, label names of fuzzy sets (linguistic terms) have appropriate names – for 2 there are 2 linguistic terms ("small" and "big"), for 3 there are 3 linguistic terms ("small", "medium", "big").

### 5.2.1. *Iris* dataset

Considering the *iris* dataset, the highest accuracy was reached by the neuro-fuzzy system NEFCLASS (Tab. 1).

[2] http://archive.ics.uci.edu/ml/

**Table 1**
Classification results for *iris* dataset

| Classifier | Accuracy [%] | Running time [s] |
|---|---|---|
| NEFCLASS | 94.59 ± 3.2 | 0.6 |
| MLP network | 92.16 ± 7.2 | 1.16 |
| k-NN | 93.78 ± 1.74 | 0.08 |

We can see, that 3 rules are sufficient to obtain good accuracy for *iris*. In the Table 2 rules generated by NEFCLASS system were shown. It can be seen that 1 attribute is adequate to classify *iris* dataset with accuracy about 94%.

**Table 2**
Rules form from NEFCLASS system for *iris* dataset

| | | | | | | |
|---|---|---|---|---|---|---|
| if | (?, | ?, | ?, | small) | then | 1 |
| if | (?, | ?, | ?, | medium) | then | 2 |
| if | (?, | ?, | ?, | big) | then | 3 |

Using of ensemble classifiers (consisting of 3 neuro-fuzzy classifiers) further increases accuracy (see Tab. 3). Both considered hybrid methods (bagging and SAMME boosting) give similar results on this dataset.

**Table 3**
Classifier results for enseble classifier consisted of NEFCLASS classifiers on *iris* dataset

| Classifier | Accuracy [%] | Running time [s] |
|---|---|---|
| Bagging | 95.41 ± 2.43 | 3.9 |
| SAMME Boosting | 95.94 ± 2.76 | 4.5 |

### 5.2.2. *Wine* dataset

On this dataset neuro-fuzzy model – NEFCLASS, gives slightly less accurate results than other presented classifiers (see Tab. 4). The algorithm works also longer than others.

**Table 4**
Classifier results for *wine* dataset

| Classifier | Accuracy [%] | Running time [s] |
|---|---|---|
| NEFCLASS | 88.63 ± 5.47 | 4.04 |
| MLP network | 97.95 ± 2.14 | 1.73 |
| k-NN | 94.77 ± 3.52 | 0.09 |

*Wine* dataset, after intensive rules pruning, can be classified based on 3 attributes (original dataset contains 13 attributes). Table 5 contains 8 rules, that are output of neuro-fuzzy model – NEFCLASS.

**Table 5**
Rules form from NEFCLASS system for *wine* dataset

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| if | (?, | ?, | ?, | ?, | ?, | ?, | big, | ?, | ?, | ?, | ?, | ?, | medium) | then | 1 |
| if | (?, | ?, | ?, | ?, | ?, | ?, | ?, | ?, | ?, | ?, | ?, | ?, | big) | then | 1 |
| if | (?, | ?, | ?, | ?, | ?, | ?, | big, | ?, | ?, | ?, | ?, | ?, | small) | then | 2 |
| if | (?, | ?, | ?, | ?, | ?, | ?, | ?, | ?, | ?, | small, | ?, | ?, | ?) | then | 2 |
| if | (?, | ?, | ?, | ?, | ?, | ?, | medium, | ?, | ?, | ?, | ?, | ?, | small) | then | 2 |
| if | (?, | ?, | ?, | ?, | ?, | ?, | small, | ?, | ?, | ?, | ?, | ?, | small) | then | 3 |
| if | (?, | ?, | ?, | ?, | ?, | ?, | small, | ?, | ?, | medium, | ?, | ?, | medium) | then | 3 |
| if | (?, | ?, | ?, | ?, | ?, | ?, | small, | ?, | ?, | big, | ?, | ?, | medium) | then | 3 |

Use of boosting methods (ensemble classifiers: bagging and SAMME boosting) improves classification accuracy (Tab. 6). In this case difference in results between these 2 methods is more than in results for *iris* dataset.

**Table 6**
Classification results for ensemble classifier consisted of NEFCLASS classifiers for *wine* dataset

| Classifier | Accuracy [%] | Running time [s] |
|---|---|---|
| Bagging | 91.13 ± 3.86 | 39.7 |
| SAMME Boosting | 92.04 ± 3.09 | 40.09 |

### 5.2.3. *Breast cancer wisconsin* dataset

For *breast cancer wisconsin* dataset, classification accuracy of NEFCLASS model is lower than in case of MLP neural networks or $k$-NN (Tab. 7).

**Table 7**
Classification results for *breast cancer wisconsin* dataset

| Classifier | Accuracy [%] | Running time [s] |
|---|---|---|
| NEFCLASS | 91.11 ± 4.05 | 13.9 |
| MLP network | 96.58 ± 1.01 | 4.34 |
| k-NN | 95.47 ± 1.34 | 0.22 |

In every step of learning algorithm of NEFCLASS classifier, rule pruning step is conducted. In Figure 2 the decrease of rules number during execution of algorithm may is shown (number of rules is presented on $y$ axis, number of algorithm steps – $x$ axis).

Presented result of neuro-fuzzy model execution consists of 3 rules (Tab. 8). These rules use 2 attributes. Use of these rules lets to achieve accuracy above 90%.

Using of ensemble classifiers again improves accuracy of classification. Moreover, boosting had better accuracy that bagging, see Table 9.

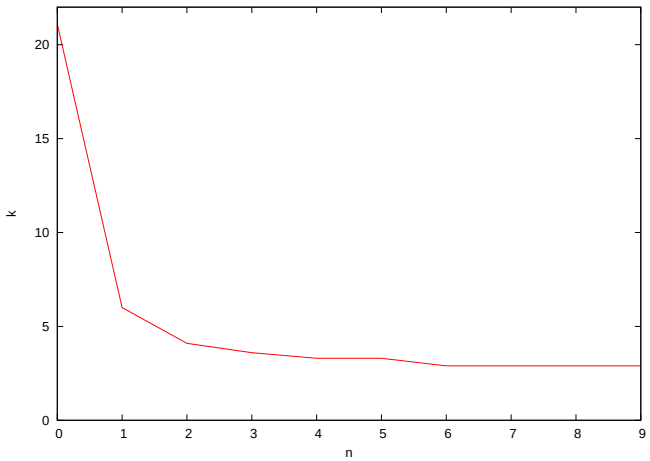


**Fig. 2.** Number of rules for *breast cancer wisconsin* dataset generated by NEFCLASS

**Table 8**
Rules form from NEFCLASS system for *breast cancer wisconsin* dataset

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| if | (?, | ?, | ?, | ?, | ?, | small, | ?, | ?, | ?) | then | 1 |
| if | (?, | ?, | ?, | ?, | ?, | ?, | ?, | ?, | big) | then | 2 |
| if | (?, | ?, | ?, | ?, | ?, | big, | ?, | ?, | ?) | then | 2 |

**Table 9**
Classification results for ensemble classifier consisted of NEFCLASS classifiers NEFCLASS for *breast cancer wisconsin* dataset

| Classifier | Accuracy [%] | Running time [s] |
|---|---|---|
| Bagging | 93.35 ± 1.66 | 52.9 |
| SAMME Boosting | 95.0 ± 1.35 | 66 |

### 5.2.4. Comparison of results for given datasets

Considering the presented results it can be seen that NEFCLASS classifier achieved the best quality for *iris* dataset and the worst for *wine* dataset. However, the results for each dataset are acceptable. It is to note, that the main advantage of NEFCLASS is the interpretability of the results being the set of rules that are readable to human.

Use of bagging and boosting methods enhances classification accuracy – for datasets with low number of instances, both methods obtain similar results, for datasets with high number of instances – boosting gains an advantage over bagging in terms of accuracy, what was visible e.g. for *breast cancer wisconsin* dataset (which consist of the most amount of instances among tested datasets). Such behavior of bagging and boosting is described in paper [26]. However, use of boosting or bagging leads to losing the interpretability feature.

## 6. Impact classifier parameters on classification accuracy

In this subsection the impact following parameters on classification accuracy will be presented:

- acceptable difference in accuracy during rule pruning,
- learning type.

Because of the fact that NEFCLASS classifier proved as the best for *iris* and the worst for *wine* dataset, to test impact parameters on its results, the *breast cancer wisconsin* dataset was chosen.

### 6.1. Acceptable difference in accuracy during rule pruning

Acceptable difference in accuracy during rule pruning means number of percent points that the result can be worse by in terms of accuracy to accept pruning.

In Table 10 the results of classifier have been located depending on this parameter. It is worth noting that classifier that can accept little worse result than before pruning, it can be more accurate. However, too high value in percentage points affects in negative way on its accuracy. Classifier achieved the best results if the value of acceptable diference was equal to 2%. With increasing of this parameter, the classifier was becoming less and less stable – the standard deviation of the average was high.

**Table 10**
Impact parameter of acceptable difference in accuracy during rule pruning on classification accuracy

| Acceptable difference | Accuracy[%] | Mean number of rules |
|---|---|---|
| 0 | 82.31 ± 5.5 | 2.4 ± 2.2 |
| 1 | 90.1 ± 4.72 | 2.6 ± 1.68 |
| 2 | 92.35 ± 3.37 | 3.2 ± 1.1 |
| 3 | 88.41 ± 6.14 | 2.4 ± 0.8 |
| 4 | 89.28 ± 3.92 | 2.3 ± 0.9 |
| 5 | 86.17 ± 17.64 | 2.6 ± 0.8 |
| 10 | 78.29 ± 11.11 | 2.1 ± 1.04 |

### 6.2. Learning type

In NEFCLASS system the several types of structural learning are used. In this subsection the results of learning on *breast cancer wisconsin* dataset by means proposed by model authors will be presented:

- simple selection,
- best rule selection,
- best per class rule selection,

and new proposed methods:

- selection rules, that have max value of activation, proportionally to number of instances in class in learning dataset,
- selection rules without defining max value of them – the number of them decrease gradually by usage of rule pruning operation.

will be compared with the original ones.

### 6.2.1. Simple selection

In this method the first $k$ rules are selected from the rules generated by structural learning model of NEFCLASS. The result of learning is presented in Table 11. Accuracy is relatively high – if we choose more rules, accuracy is better. Pruning of rules is done every 5 step of processing. One of disadvantage of this learning type is high deviation of the results of classifier – it is caused by the fact that first $k$ rules may vary significantly.

**Table 11**
Result of simple rules selection for NEFCLASS on *breast cancer wisconsin* dataset

| Max number of rules | Accuracy [%] | Mean final number of rules |
|---|---|---|
| 5 | 82.11 ± 12.5 | 1.5 ± 0.5 |
| 10 | 86.71 ± 6.17 | 2.3 ± 0.9 |

### 6.2.2. Best rule selection

In this type of learning $k$ rules, that have highest activation value, are chosen to the set of rules. Results obtained by using this method (Tab. 12) are better than obtained by using the previous one. Authors of NEFCLASS model (Nauck and Kruse) recommend using this method because this learning type often achieve best results on majority of datasets.

**Table 12**
Results of best rule selection for NEFCLASS on *breast cancer wisconsin* dataset

| Max number of rules | Accuracy [%] | Mean final number of rules |
|---|---|---|
| 5 | 86.52 ± 2.63 | 1.5 ± 0.5 |
| 10 | 87.82 ± 2.51 | 2.0 |

### 6.2.3. Best per class rule selection

During this learning method the same number of rules is assigned to each class, for each class rules with best activation value of rule for given class. Result obtained by selection of 10 rules (Tab. 13) is near to result achieved by selection of 10 rules with use of previous method.

**Table 13**
Result of best per class rule selection for NEFCLASS on *breast cancer wisconsin* dataset

| Max number of rules | Accuracy [%] | Mean final number of rules |
|---|---|---|
| 5 | 83.82 ± 4.95 | 1.8 ± 0.4 |
| 10 | 87.76 ± 3.37 | 2.1 ± 0.83 |

### 6.2.4. Proportional best per class rule selection

This method is similar to prior method of learning, the one difference is that in the former one, for each class the same number of rules was chosen, whilst in this method the number of rules for each class is predefined according to number of instances in each class. The results are shown in Table 14 and are near to results obtained by method of best rule selection.

**Table 14**
Results of proportional best rules selection for NEFCLASS on *breast cancer wisconsin* dataset

| Max number of rules | Accuracy [%] | Mean final number of rules |
|---|---|---|
| 5 | 86.64 ± 3.09 | 1.9 ± 0.4 |
| 10 | 88.23 ± 5.68 | 2.0 |

### 6.2.5. Rule selection with no rule limit number

In this method there is no upper limit for rules. Therefore, in the beginning, large number of rules is selected to rule set, but during learning their number it gradually decreases. For this method the obtained results are best among results of other learning types for this dataset. Intensive rule pruning is conducted in every step and gives better effect than rule pruning every 5th step – see Tab. 15. However, in that situation more rules are generated than during learning of other methods.

**Table 15**
Results of selection rules with no rule limit number for NEFCLASS on *breast cancer wisconsin* dataset

| Rule pruning | Accuracy [%] | Mean final number of rules |
|---|---|---|
| every step | 92.23 ± 2.41 | 4.2 ± 2.04 |
| every 5 step | 91.82 ± 2.14 | 5.6 ± 2.41 |

### 6.2.6. Summary of different types of learning

To summarize experiments, best results from discussed methods are obtained with use no rule limit selection of rules. During algorithm execution, quantity of rules is decreased in each step. Other methods also present good accuracy – the best from them is best rule selection.

## 7. Conclusions

We presented the classifier based on neuro-fuzzy model – NEFCLASS, which was modified in relation to original version of that model. There are proposed 2 new training methods and some heuristic methods of pruning rules. Modified model achieved better results than neural networks and $k$-nearest neighbours classifier on the iris dataset, on 2 remaining datasets (the wine dataset and the breast cancer wisconsin original dataset) the results were slightly worse. The parameters in tested classifiers (including implemented classifier) were not tuned for each dataset. What is worth noting – the NEFCLASS model is oriented on interpretability, not accuracy.

The results, described above, show that classifier is effective on tested datasets. Its accuracy can be improved by using ensemble methods but in effect the one of advantages of this classifier is lost – the interpretability. In ensemble methods 2 groups can be highlighted: bagging and boosting. The larger dataset is, the greater growth of accuracy boosting causes than bagging what is described by reseachers [26].

In the future we plan to focus on applying the classifier to some real-world problems (besides testing benchmark problems) and incorporate it in other soft-computing systems in the form of software component.

## References

[1] Breiman L. *Bagging predictors.* Machine Learning, 24, 1996, pp. 123–140.

[2] Burges C.: *A tutorial on support vector machines for pattern recognition.* Data Mining and Knowledge Discovery, 2, 1998, pp. 121–167.

[3] Burges C., Scholkopf B.: *Improving the accuracy and speed of support vector machines.* Neural Information Processing Systems, 9, 1997.

[4] Dong-Sheng Cao, Qing-Song Xu, Yi-Zeng Liang, Liang-Xiao Zhang, Hong-Dong Li: *The boosting: A new idea of building models.* Chemometrics and Intelligent Laboratory Systems, 100, 2010.

[5] Carroll J. L.: *No free-lunch and bayesian optimality.* [in:] *IJCNN Workshop on Meta-Learning*, 2007.

[6] Cichosz P.: *Systemy uczące się.* Wydawnictwa Naukowo-Techniczne, Warszawa, 2000.

[7] Fowler M.: *Inversion of control containers and the dependency injection pattern.* `http://www.martinfowler.com/articles/injection.html`.

[8] Freund Y., Schapire R. E.: *Experiments with a new boosting algorithm.* [in:] Machine Learning: Proceedings of the Thirteenth International Conference, 1996.

[9] Haykin S.: *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 1999.

[10] Klawonn F., Kruse R.: *Constructing a fuzzy controller from data.* Fuzzy Sets and Systems, 85, 1997, pp. 177–193.

[11] Klawonn F., Nauck. D.: *Automatically determine initial fuzzy partitions for neuro–fuzzy classifiers.* [in:] 2006 IEEE International Conference on Fuzzy Systems, 2006, pp. 1703–1709.

[12] Koronacki J., Ćwik J.: *Statystyczne systemy uczące się.* Akademicka Oficyna Wydawnicza EXIT, 2nd ed., 9 2008.

[13] Krzyśko M., Wołyński W., Górecki T., Skorzybut M.: *Statystyczne systemy uczące się. Rozpoznawanie wzorców, analiza skupień i redukcja wymiarowości.* Wydawnictwa Naukowo-Techniczne, Warszawa, 2008.

[14] Ligeza A.: *Logical Foundations for Rule-Based Systems.* Springer, 2006.

[15] Łęski J.: *Systemy neuronowo-rozmyte.* Wydawnictwa Naukowo-Techniczne, 2008.

[16] Mamdani E. H., Assilian S.: *An experiment in linguistic synthesis with a fuzzy logic controller.* Journal of Man-Machine Studies, 7(1), 1975, pp. 1–13.

[17] Mitra S., Pal S. K.: *Fuzzy sets in pattern recognition and machine intelligence.* Fuzzy Sets and Systems, 156, 2005.

[18] Nauck D., Nauck U., Kruse R.: *Generating classification rules with the neuro-fuzzy system NEFCLASS.* [in:] 1996 Biennial Conference of the North American, Fuzzy Information Processing Society, 1996, pp. 466–470.

[19] Nauck D., Kruse R.: *Nefclass – a neuro-fuzzy approach for the classification of data.* [in:] Applied Computing 1995. Proc. of the 1995 ACM Symposium on Applied Computing, ACM Press, 1995, pp. 461–465.

[20] Nauck D., Kruse R.: *How the learning of rule weights affects the interpretability of fuzzy systems.* [in:] Proc. IEEE International Conference on Fuzzy Systems, Anchorage, 1998, pp. 1235–1240.

[21] Nauck D., Kruse R.: *Obtaining interpretable fuzzy classification rules from medical data.* Artificial Intelligence in Medicine, 16, 1999.

[22] Nauck D. D.: *Fuzzy data analysis with NEFCLASS.* International Journal of Approximate Reasoning, 32, 2003.

[23] Pedrycz W., Gomide F.: *An Introduction to Fuzzy Sets: Analysis and Design.* The MIT Press, 1998.

[24] Rokach L.: *Ensemble-based classifiers.* Artificial Intelligence Review, 33(1–2), 2010, pp. 1–39.

[25] Rutkowski L.: *Flexible Neuro-Fuzzy Systems.* Kluwer, 2004.

[26] Skurichina M., Kuncheva L. I., Duin R. P. W.: *Bagging and boosting for the nearest mean classifier: Effects of sample size on diversity and accuracy. Multiple classifier systems: Third International Workshop*, MCS 2002, 2364, 2002, pp. 62–71.

[27] Tadeusiewicz R.: *Sieci neuronowe.* Akademicka Oficyna Wydaw. RM, Warszawa, 1993.

[28] Xindong Wu, Kumar V., Quinlan J. R., Ghosh J., Yang Q., Motoda H., McLachlan G. J., Ng A., Bing Liu, Yu P. S., Zhi-Hua Zhou, Steinbach I., Hand D. J., Steinberg D.: *Top 10 algorithms in data mining.* Knowledge and Information Systems, 14(1), 2008, pp. 1–37.

[29] Ji Zhu, Rosset S., Hui Zou, Hastie T.: *Multi-class AdaBoost.* Statistics and its interface, 2, 2009, pp. 349–360.