Eliminators and classifiers

Włodzisław Duch and Rafał Adamczak

Department of Computer Methods
Nicholas Copernicus University
Grudziądzka 5, 87-100 Toruń, Poland.
WWW: http://www.phys.uni.torun.pl/∼duch

Yoichi Hayashi

Department of Computer Science
Meiji University, Kawasaki 214-8571, Japan
Email: hayashiy@cs.meiji.ac.jp

Abstract

Classification may not be reliable for several reasons: noise in the data, insufficient input information, overlapping distributions and sharp definition of classes. Faced with K possibilities a decision support system may still be useful in such cases if instead of classification elimination of improbable classes is done. Eliminators may be constructed using classifiers assigning new cases to a pool of several classes instead of just one winning class. Elimination may be done with the help of several classifiers using modified error functions. A real life medical example is presented illustrating the usefulness of elimination.

1 Introduction.

Neural, fuzzy and machine learning systems are usually applied as classifiers or approximators. In real-world problems designation of classes may be problematic. One of the reasons may be due to the approximate nature of linguistic concepts that are used to label cases changing in a continuous way. Medical databases contain names of diseases which may develop in time, from mild to severe cases, with intermediate or mixed forms, and thus giving rise to strongly overlapping class distributions $p(\mathbf{X}|C_i)$ and fuzzy labels. The information provided in the database may be insufficient to distinguish the classes which may be well differentiated by some unknown features (for example, results of a better medical test). In such situations reliable classification is not possible and comparison of results using the accuracy based on the number of classification errors may be quite misleading.

If soft class labels are needed or if insufficient number of classes are defined some conclusions can still be drawn by looking at the classification probabilities. For example, the system may assign an unknown case to the overlapping region where two or more classification probabilities have significant values, in a way creating new, mixed or border classes. Introduction of new classes cannot be done automatically and requires close collaboration with the domain expert. An alternative way of solving such problems is to eliminate improbable classes, predicting that the unknown case belongs to a subset of k classes out of K possible ones. The number k should not be fixed to account for the possibility of class distributions overlapping in a different way in different regions of the input space. Such systems may be called **eliminators** since their primary goal is to eliminate with high confidence classes that are improbable.

Any model M that estimates probabilities of classification $p(C_i|\mathbf{X};M)$ may be used to create new, soft class labels or to eliminate some classes predicting that $\mathbf{X}$ belongs to two or more classes. In particular neural and neuro-fuzzy systems are well-suited for this purpose, although they should be modified to optimize elimination of classes rather then predicting one class. Some other classification systems, such as statistical discrimination methods, support vector machines [1], decision trees or the nearest neighbor methods provide only sharp yes/no classification decisions [2]. Detailed interpretation of a given case is possible if methods of explanatory data analysis attempting to visualize the case in relation to known cases from the training database are used or if classification confidence intervals are calculated [3].

Our goal in this paper is two-fold. In the next section problems specific to class elimination in neural networks are disussed, followed by presentation of a universal method for estimation of probabilities that is applicable to any classifier. A real-life example of a difficult medical problem is presented in the fourth section and a short discussion concludes this paper.

2 Elimination instead of prediction

Consider a classification problem in N dimensions with two overlapping classes described by Gaussian distributions with equal covariance matrices Σ:

$$p(\mathbf{X}|C_i) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{X}-\bar{\mathbf{X}}_i)^T\Sigma^{-1}(\mathbf{X}-\bar{\mathbf{X}}_i)\right\}$$

Using Bayes' theorem the posterior probability for the

first class is [4]:

$$p(C_1|\mathbf{X}) = \frac{p(\mathbf{X}|C_1)P(C_1)}{p(\mathbf{X}|C_1)P(C_1) + p(\mathbf{X}|C_2)P(C_2)} \quad (1)$$

The $P(C_k)$ are *a priori* class probabilities. Thus $p(C_1|\mathbf{X}) = \sigma(y(\mathbf{X}))$, where the function $y(\mathbf{X})$ is:

$$y(\mathbf{X}) = \ln \frac{p(\mathbf{X}|C_1)P(C_1)}{p(\mathbf{X}|C_2)P(C_2)} = \mathbf{W}\cdot\mathbf{X} - \theta \quad (2)$$

where

$$\mathbf{W} = (\mathbf{X}_2 - \mathbf{X}_1)^T \Sigma^{-1} = \mathbf{W}\cdot\mathbf{X} - \theta \quad (3)$$

and $\theta = \theta(\mathbf{X}_1, \mathbf{X}_2, \Sigma, P(C_1), P(C_2))$. The posterior probability is thus given by a specific logistic output function. For more than two classes normalized exponential functions (called also softmax functions) are obtained by the same reasoning:

$$p(C_k|\mathbf{X}) = \frac{\exp(y_k(\mathbf{X}))}{\sum_i \exp(y_i(\mathbf{X}))} \quad (4)$$

These normalized exponential functions may be interpreted as probabilities. They are provided in a natural way by multilayer perceptron networks (MLPs). If one of the probabilities is close to 1 or to 0 the situation is clear. Otherwise $\mathbf{X}$ belongs to the border area and a unique classification may not be possible. The domain expert should decide if it makes sense to introduce a new, mixed class, or to acknowledge that insufficient information is available for accurate classification.

2.1 Measures of classifier performance

Measures of classifier performance based on accuracy of confusion matrices $F(C_i, C_j)$ do not allow to evaluate their usefulness. Introduction of risk matrices or use of receiver-operator characteristic (ROC) curves [5] does not solve the problem either.

If the standard approach fails to provide sufficiently accurate results for some classes one should either attempt to create new classes or to minimize the number of errors between a temporary new class composed of two or more distinct classes. This requires a modification of the standard cost function. Let $C(\mathbf{X};M)$ be the class predicted by the model M and $p(C_i|\mathbf{X};M)$ the probability of class C_i. In most cases the cost function minimizes the classification error and has either a quadratic form:

$$E_2(\{\mathbf{X}\}, R; M) = \sum_i \sum_{\mathbf{X}} \left(p(C_i|\mathbf{X}) - \delta(C_i, C(\mathbf{X}))\right)^2 \quad (5)$$

or – since continuos output values are provided by some models – minimization of risk for overall classification:

$$E(\{\mathbf{X}\}, R; M) = \sum_i \sum_{\mathbf{X}} R(C_i, C(\mathbf{X})) H\left(p(C_i|\mathbf{X};M) - \delta(C_i, C(\mathbf{X}))\right) \quad (6)$$

where i runs over all different classes and $\mathbf{X}$ over all training vectors, $C(\mathbf{X})$ is the true class of the vector $\mathbf{X}^p$ and the function $H(\cdot)$ should be monotonic and positive; quite often the quadratic function or entropy-based function are used.

The elements of the risk matrix $R(C_i, C_j)$ are proportional to the risk of assigning the C_i class when the true class is C_j (in the simplest case $R(C_i, C_j) = 1 - \delta_{ij}$), and M specifies all adaptive parameters and variable procedures of the classification model that may affect the cost function. Regularization terms aimed at minimization of the complexity of the classification model are frequently added to the cost function, allowing to avoid the overfitting problems. To minimize the leave-one-out error the sum runs over all training examples $\mathbf{X}^p$ and the model used to specify the classifier should not contain the $\mathbf{X}^p$ vector in the reference set while $p(C_i|\mathbf{X}^p)$ is computed.

A simplified version of the cost function is also useful:

$$\begin{aligned} C_j(\mathbf{X}^p) \leftarrow j &= \arg\max_i \; p(C_i|\mathbf{X}^p;M) \\ E(\{X\};M) &= \sum_p K\left(C(\mathbf{X}^p) - C_j(\mathbf{X}^p)\right) \end{aligned} \quad (7)$$

where $C_j(\mathbf{X}^p)$ corresponds to the best recommendation of the classifier and the kernel function $K(\cdot,\cdot)$ measures similarity of the classes. A general expression is:

$$E\left(\{X\};M\right) = \sum_i K\left(d\left(X^{(i)}, R\right)\right) Err\left(X^{(i)}\right) \quad (8)$$

For example in the local regression based on the minimal distance approaches [6] the error function is:

$$E(\{X\};M) = \sum_p K(D(\mathbf{X}^p, \mathbf{X}^{ref}))(F(\mathbf{X}^p;M) - y^p)^2 \quad (9)$$

where y^i are the desired values for $\mathbf{X}^i$ and $F(\mathbf{X}^i;M)$ are the values predicted by the model M. Here the kernel function $K(d)$ measures the influence of the reference vectors on the total error. For example, if $K(d)$ has a sharp high peak around $d = 0$ the function $F(\mathbf{X};M)$ will fit the values corresponding to the reference input vectors almost exactly and will make large errors for other values. In classification problems kernel function will determine the size of the neighborhood around the known cases in which accurate classification is required.

Suppose that both off-diagonal elements F_{12} and F_{21} of the confusion matrix are large, i.e. the two classes are frequently mixed. In this case we can try to separate these two classes from all the others using an independent classifier. The joint class is designated $C_{1,2}$ and the model trained with the following cost function:

$$E_d(\{X\};M) = \sum_{\mathbf{X}} H\left(p(C_{1,2}|\mathbf{X};M) - \delta(C_{1,2},C(\mathbf{X}))\right) + \sum_{k>2}\sum_{\mathbf{X}} H\left(p(C_k|\mathbf{X};M) - \delta(C_k,C(\mathbf{X}))\right) \quad (10)$$

Training with such error function provides new, possibly simpler, decision borders. In practice one should use classifier first and only if classification is not sufficiently reliable (several probabilities are almost equal) try to eliminate subsets of classes. If joining pairs of classes is not sufficient triples and higher combinations may be considered.

3 Calculation of probabilities

Some classifiers do not provide probabilities, therefore it is not clear how to optimized them for elimination of classes instead of selection of the most probable class. A universal solution independent of any classifier system is described below.

Real input values $\mathbf{X}$ are obtained by measurements that are carried with finite precision. The brain uses not only large receptive fields for categorization, but also small receptive fields to extract feature values. Instead of a crisp number X a Gaussian distribution $G_X = G(Y;X,S_X)$ centered around X with dispersion S_X should be used. Probabilities $p(C_i|\mathbf{X};M)$ may be computed for any classification model M by performing a Monte Carlo sampling from the joint Gaussian distribution for all continuous features $G_\mathbf{X} = G(\mathbf{Y};\mathbf{X},\mathbf{S}_X)$. Dispersions $\mathbf{S}_X = (s(X_1),s(X_2)\ldots s(X_N))$ define the volume of the input space around $\mathbf{X}$ that has an influence on computed probabilities. One way to "explore the neighborhood" of $\mathbf{X}$ and see the probabilities of alternative classes is to increase the fuzziness $\mathbf{S}_X$ defining $s(X_i) = (X_{i,max} - X_{i,min})\rho$, where the parameter ρ defines a percentage of fuzziness relatively to the range of X_i values.

With increasing ρ values the probabilities $p(C_i|\mathbf{X};\rho,M)$ change. Even if a crisp rule-based classifier is used nonzero probabilities of classes alternative to the winning class will gradually appear. The way in which these probabilities change shows how reliable is the classification and what are the alternatives worth remembering. If the probability $p(C_i|\mathbf{X};\rho,M)$ changes rapidly around some value ρ_0 the case $\mathbf{X}$ is near classification border and an analysis of $p(C_i|\mathbf{X};\rho_0,s_i,M)$ as a function of each $s_i = s(X_i), i = 1\ldots N$ is needed to see which features have strong influence on classification. Displaying such probabilities allows for a detailed evaluation of the new data also in cases where analysis of rules is too complicated. A more detailed analysis of these probabilities based on *confidence intervals* and *probabilistic confidence intervals* has recently been presented by Jankowski [7]. Confidence intervals are calculated individually for a given input vector while logical rules are extracted for the whole *training set*. Confidence intervals measure maximal deviation from the given feature value X_i (assuming that other features of the vector $\mathbf{X}$ are fixed) that do not change the most probable classification of the vector $\mathbf{X}$. If this vector lies near the class border the confidence intervals are narrow, while for vectors that are typical for their class confidence intervals should be wide. These intervals facilitate precise interpretation and allow to analyze the stability of sets of rules.

For some classification models probabilities $p(C_i|\mathbf{X};\rho,M)$ may be calculated analytically. For the crisp rule classifiers [8] a rule $R_{[a,b]}(X)$, which is true if $X \in [a,b]$ and false otherwise, is fulfilled by a Gaussian number G_X with probability:

$$p(R_{[a,b]}(G_X) = T) \approx \sigma(\beta(X-a)) - \sigma(\beta(X-b)) \quad (11)$$

where the logistic function $\sigma(\beta X) = 1/(1+\exp(-\beta X))$ has $\beta = 2.4/\sqrt{2}s(X)$ slope. For large uncertainty $s(X)$ this probability is significantly different from zero well outside the interval $[a,b]$. Thus crisp logical rules for data with Gaussian distribution of errors are equivalent to fuzzy rules with "soft trapezoid" membership functions defined by the difference of the two sigmoids, used with crisp input value. The slope of these membership functions, determined by the parameter β, is inversely proportional to the uncertainty of the inputs. In the C-MLP2LN neural model [9] such membership functions are computed by the network "linguistic units" $L(X;a,b) = \sigma(\beta(X-a)) - \sigma(\beta(X-b))$. Relating the slope β to the input uncertainty allows to calculate probabilities in agreement with the Monte Carlo sampling. Another way of calculating probabilities, based on the softmax neural outputs $p(C_j|\mathbf{X};M) = O_j(\mathbf{X})/\sum_i O_i(\mathbf{X})$ has been presented in [7].

After uncertainty of inputs has been taken into account probabilities $p(C_i|\mathbf{X};M)$ depend in a continuous way on intervals defining linguistic variables. The error function:

$$E(M,\mathbf{S}) = \frac{1}{2}\sum_{\mathbf{X}}\sum_i \left(p(C_i|\mathbf{X};M) - \delta(C(\mathbf{X}),C_i)\right)^2 \quad (12)$$

depends also on uncertainties of inputs $\mathbf{S}$. Several variants of such models may be considered, with Gaussian or conical (triangular-shaped) assumptions for input distributions, or neural models with bicentral transfer functions in

the first hidden layer. Confusion matrix computed using probabilities instead of the number of yes/no errors allows for optimization of the error function using gradient-based methods. This minimization may be performed directly or may be presented as a neural network problem with a special network architecture. Uncertainties s_i of the values of features may be treated as additional adaptive parameters for optimization. To avoid too many new adaptive parameters optimization of all, or perhaps of a few groups of s_i uncertainties, is replaced by common ρ factors defining the percentage of assumed uncertainty for each group.

This approach leads to the following important improvements for any rule-based system:

- Crisp logical rules provide basic description of the data, giving maximal comprehensibility.
- Instead of 0/1 decisions probabilities of classes $p(C_i|\mathbf{X};M)$ are obtained.
- Inexpensive gradient method are used allowing for optimization of very large sets of rules.
- Uncertainties of inputs s_i provide additional adaptive parameters.
- Rules with wider classification margins are obtained, overcoming the brittleness problem of some rule-based systems.

Wide classification margins are desirable to optimize the placement of decision borders, improving generalization of the system. If the vector $\mathbf{X}$ of an unknown class is quite typical to one of the classes C_k increasing uncertainties s_i of X_i inputs to a reasonable value (several times the real uncertainty, estimated for a given data) should not decrease the $p(C_k|\mathbf{X};M)$ probability significantly. If this is not the case $\mathbf{X}$ may be close to the class border and analysis of $p(C_i|\mathbf{X};\rho,s_i,M)$ as a function of each s_i is needed. These probabilities allow to evaluate the influence of different features on classification. If simple rules are available such explanation may be satisfactory. Otherwise to gain understanding of the whole data a similarity-based approach to classification and explanation is worth trying. Prototype vectors R_i are constructed using a clusterization, dendrogram or a decision tree algorithm and a similarity measure $D(\mathbf{X},\mathbf{R})$ is introduced. Positions of the prototype vectors R_i, parameters of the similarity measures $D(\cdot)$ and other adaptive parameters of the system are then optimized using a general framework for similarity-based methods [10]. This approach includes radial basis function networks, clusterization procedures, vector quantization methods and generalized nearest neighbor methods as special examples. An explanation in this case is given by pointing out to the similarity of the new case $\mathbf{X}$ to one or more of the prototype cases R_i.

Similar result is obtained if the linear discrimination analysis (LDA) is used - instead of sharp decision border in the direction perpendicular to LDA hyperplane a soft logistic function is used, corresponding to a neural network with singe neuron. The weights and bias are fixed by the LDA solution, only the slope of the function is optimized.

4 Real-life example

Hepatobiliary disorders data, used previously in several studies [17, 11, 12, 16], contains medical records of 536 patients admitted to a university affiliated Tokyo-based hospital, with four types of hepatobiliary disorders: alcoholic liver damage (AL), primary hepatoma (PH), liver cirrhosis (LC) and cholelithiasis (CH). The records includes results of 9 biochemical tests and sex of the patient. The same 163 cases as in [17] were used as the test data.

In the previous work three fuzzy sets per each input were assigned using recommendation of the medical experts. A fuzzy neural network was constructed and trained until 100% correct answers were obtained on the training set. The accuracy on the test set varied from less than 60% to a peak of 75.5%. Although we quote this result in the Table 1 below it seems impossible to find good criteria that will predict when the training on the test set should be stopped. Fuzzy rules equivalent to the fuzzy network were derived but their accuracy on the test set was not given. This data has also been analyzed by Mitra et al. [18, 16] using a knowledge-based fuzzy MLP system with results on the test set in the range from 33% to 66.3%, depending on the actual fuzzy model used.

For this dataset classification using crisp rules was not too successful. The initial 49 rules obtained by C-MLP2LN procedure gave 83.5% on the training and 63.2% on the test set. Optimization did not improve these results significantly. On the other hand fuzzy rules derived using the FSM network, with Gaussian as well as with triangular functions, gave similar accuracy of 75.6-75.8%. Fuzzy neural network used over 100 neurons to achieve 75.5% accuracy, indicating that good decision borders in this case are quite complex and many logical rules will be required. Various results for this dataset are summarized in Table 1.

FSM gives about 60 Gaussian or triangular membership functions achieving accuracy of 75.5-75.8%. Rotation of these functions (i.e. introducing linear combination of inputs to the rules) does not improve this accuracy. We have also made 10-fold crossvalidation tests on the mixed data (training plus test data), achieving similar results. Many methods give rather poor results on this dataset, including various variants of the instance-based learning (IB2-IB4, except for the IB1c, which is specifically designed to work with continuous input data), statistical methods

Table 1: Results for the hepatobiliary disorders. Accuracy on the training and test sets, in %. Top results are achieved eliminating classes or predicting pairs of classes. All calculations are ours except where noted.

Method	Training set	Test set
FSM-50, 2 most prob. classes	96.0	92.0
FSM-50, class 2+3 combined	96.0	87.7
FSM-50, class 1+2 combined	95.4	86.5
Neurorule [11]	85.8	85.6
Neurolinear [11]	86.8	84.6
1-NN, weighted (ASA)	83.4	82.8
FSM, 50 networks	94.1	81.0
1-NN, 4 features	76.9	80.4
K* method	–	78.5
kNN, k=1, Manhattan	79.1	77.9
FSM, Gaussian functions	93	75.6
FSM, 60 triangular functions	93	75.8
IB1c (instance-based)	–	76.7
C4.5 decision tree	94.4	75.5
Fuzzy neural network [16, 18]	100	75.5
Cascade Correlation	–	71.0
MLP with RPROP	–	68.0
Best fuzzy MLP model [12]	75.5	66.3
C4.5 decision rules	64.5	66.3
DLVQ (38 nodes)	100	66.0
LDA (statistical)	68.4	65.0
49 crisp logical rules	83.5	63.2
FOIL (inductive logic)	99	60.1
T2 (rules from decision tree)	67.5	53.3
1R (rules)	58.4	50.3
Naive Bayes	–	46.6
IB2-IB4	81.2-85.5	43.6-44.6

(Bayes, LDA) and pattern recognition methods (LVQ).

The best classification results were obtained with the committee of 50 FSM neural networks [14, 15] (in Table 1 shown as FSM-50), reaching 81%. The *k*-nearest neighbors (kNN) with k=1, Manhattan distance function and selection of features gives 80.4% accuracy (for details see [13]) and after feature weighting 82.8% (the training accuracy of kNN is estimated using the leave-one-out method). K* method based on algorithmic complexity optimization gives 78.5% on the test set, with other methods giving significantly worse results.

The confusion matrix obtained on the training data from the FSM system, averaged over 5 runs and rounded to integer values is (rows - predicted, columns - required):

$$\begin{pmatrix} & AL & PH & LC & CH \\ AL & 70 & 6 & 3 & 3 \\ PH & 3 & 121 & 3 & 1 \\ LC & 1 & 8 & 77 & 2 \\ CH & 0 & 0 & 0 & 72 \end{pmatrix} \quad (13)$$

Looking at the confusion matrix one may notice that the main problem comes from predicting AL or LC when the true class is PH. The number of vectors that are classified incorrectly with high confidence (probability over 0.9) in the training data is 10 and in the test data 7 (only 4.3%). Rejection of these cases increases confidence in classification, as shown in Fig. 1.

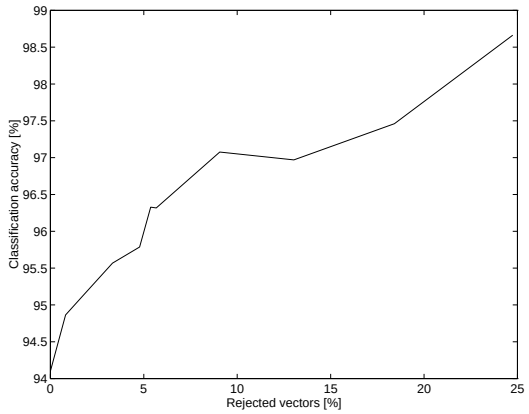

Figure 1: Relation between the accuracy of classification and the rejection rate.

In [11, 12] a "relaxed success criterion" has been used, counting as a success if the first two strongly excited output neurons contain the correct class. This is equivalent to elimination of 2 classes, leaving the combination of other two as the most probable. In this case accuracy improves, reaching about 90%. In [11] two rule extraction methods, *Neurorule* and *Neurolinear* are used, and the best test set results reach 88.3% and 90.2% respectively. Unfortunately true classification accuracy results of these methods are significantly worse then those quoted in Table 1, reaching only 48.4% (*Neurorule*) and 54.4% (*Neurolinear*) [11] on the test set. We have used here the elimination approach defining first a committee of 50 FSM networks that classify 81% of cases correctly with high reliability, while cases which cannot be reliably classified are passed to the second stage, in which elimination of pairs of classes (1+2 or 2+3) is made. Training a "supersystem" with error function given by Eq. 10 that tries to obtain the true class as one of the two most probable classes gives 92% correct answers on the test and 96% on the training set. This high accuracy unfortunately drops to 87% if a threshold of $p \geq 0.2$

is introduced for the second class. In any case we can reliably diagnose about 80% of the test cases and for the half of the remaining cases we can eliminate two classes and assign the case under consideration to a mixture of the two remaining classes.

5 Discussion

If reliable classification in a multi-class problem is impossible one can still provide a useful decision support using a classifier that is able to predict some cases with high confidence and an eliminator that can reliably eliminate several classes. The case under consideration most probably belongs to a mixture of remaining classes. Eliminators are build by analysis of confusion matrices and training classifiers with modified error functions.

Since not all classifiers provide probabilities and thus allow to estimate the confidence in their decisions we have described here a universal way to obtain probabilities $p(C_k|\mathbf{X};\rho,M)$ using Monte Carlo estimations. In some cases these probabilities may be determined analytically. Since usually only one new case is evaluated at a time (for example in medical applications) the cost of Monte Carlo simulations is not so relevant. Further research to determine the best ways of elimination of classes and applications of such approach to medical and other problems are under way.

References

[1] N. Cristianini, J. Shawe-Taylor, *"An introduction to support vector machines (and other kernel-based learning methods),"* Cambridge University Press, 2000

[2] D. Michie, D.J. Spiegelhalter, C.C. Taylor (eds.), *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, New York, 1994

[3] W. Duch, Y. Hayashi, "Computational intelligence methods and data understanding," *International Symposium on Computational Intelligence*, Kosice - Slovakia, August 2000 (in print)

[4] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[5] J.A. Swets, "Measuring the accuracy of diagnostic systems." *Science*, Vol. 240, pp. 1285-93, 1988

[6] C.G. Atkenson, A.W. Moor and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review*, Vol. 11, pp. 75-113, 1997

[7] Duch, W., Jankowski, N., Adamczak, R., Grąbczewski, K. (2000) Optimization and Interpretation of Rule-Based Classifiers. Intelligent Information Systems IX, Springer Verlag (in print)

[8] Duch, W., Adamczak, R., Grąbczewski, K. (1999) Methodology of extraction, optimization and application of logical rules. Intelligent Information Systems VIII, Ustroń, Poland, 14-18.06.1999, pp. 22-31

[9] Duch, W., Adamczak, R., Grąbczewski, K. (1998) Extraction of logical rules from backpropagation networks. Neural Processing Letters **7**, 1-9

[10] Duch, W. (1998) A framework for similarity-based classification methods, Intelligent Information Systems VII, Malbork, Poland, June 1998, pp. 288–291

[11] Y. Hayashi, R. Setiono and K. Yoshida, "A comparison between two neural network rule extraction techniques for the diagnosis of hepatobiliary disorders." Artificial Intelligence in Medicine, 2000 (in print).

[12] Y. Hayashi, R. Setiono and K. Yoshida, "Diagnosis of hepatobiliary disorders using rules extracted from artificial neural networks." In: Pro. 1999 IEEE International Fuzzy Systems Conference, Seoul, Korea, August 1999, vol. I, pp. 344-348.

[13] W. Duch, R. Adamczak, K. Grąbczewski, G. Żal, Y. Hayashi, "Fuzzy and crisp logical rule extraction methods in application to medical data." Computational Intelligence and Applications. Springer Studies in Fuzziness and Soft Computing, Vol. 23 (ed. P.S. Szczepaniak), Springer 2000, pp. 593-616

[14] Duch, W., Diercksen, G.H.F. (1995) *Feature Space Mapping as a universal adaptive system*, Computer Physics Communication **87**, 341–371

[15] Duch, W., Adamczak, R., Jankowski, N. (1997) New developments in the Feature Space Mapping model. 3rd Conf. on Neural Networks, Kule, Poland, Oct. 1997, pp. 65-70

[16] Pal, S.K. and Mitra S. (1999) *Neuro-Fuzzy Pattern Recognition*. J. Wiley, New York

[17] Y. Hayashi, A. Imura, K. Yoshida, "Fuzzy neural expert system and its application to medical diagnosis", in: 8th International Congress on Cybernetics and Systems, New York City 1990, pp. 54-61

[18] S. Mitra, R. De, S. Pal, "Knowledge based fuzzy MLP for classification and rule generation", IEEE Transactions on Neural Networks 8, 1338-1350, 1997