Prototype based rules - a new way to understand the data.

Włodzisław Duch and Karol Grudziński
Department of Computer Methods, Nicholas Copernicus University,
Grudziądzka 5, 87-100 Toruń, Poland.
WWW: http://www.phys.uni.torun.pl/kmk

Abstract.

Logical rules are not the only way to understand the structure of data. Prototype-based rules evaluate similarity to a small set of prototypes using optimized similarity measures. Such rules include crisp and fuzzy logic rules as special cases and are natural way of categorization from psychological point of view. An elimination procedure selecting good prototypes from a training set has been described. Illustrative applications on several datasets show that a few prototypes may indeed explain the data structure.

Introduction.

Knowledge discovery using neural and other computational intelligence methods is a fast-growing field [1], essential for the data mining applications. The field has concentrated on discovering logical description of the data. Some form of explanation of the data structure, or even better, theory-building, is always desired. Knowledge discovery became almost synonymous with extraction of logical rules from data. Sets of rules, if sufficiently simple and accurate, provide a very powerful explanation of the data. Crisp logical rules are the simplest and therefore most desirable for initial analysis. Fuzzy rules are natural extension of the crisp rules, increasing their expressive power. Finding the simplest logical description of data is not an easy task. It is worthwhile to look first for crisp rules and if they are insufficient for fuzzy rules. Recently we have developed several methods of logical rule extraction and analyzed many datasets, providing in most cases the simplest logical descriptions found so far [2]. However, extraction of logical rules is not the only, and sometimes not the best, way to understand the data.

What constitutes a satisfactory explanation differs from field to field and should be a matter of cognitive psychology studies. Understanding or explanation of the data may be achieved either by visualization, logical rule induction or by case-based reasoning. Visualization, called also an exploratory data analysis, is often used in medicine and in many fields of science. For example, weather maps resulting from computer simulations allow to understand meteorological data for large areas of the world in much better ways than any logical rules. Medical images allow to see changes in the parameters describing the body tissue, but these images have to be interpreted. It is doubtful that humans use prepositional logic in this process. Interpretations of images, learning the structure of irregular languages, understanding decisions of lawyers in the British legal system requires case-based rather than rule-based reasoning.

Even if logical rules may seem appropriate the complexity of the set of rules extracted from the data may be high and real understanding of the data using logical rules may not be possible. Fuzzy rules rarely offer significantly less complex description since the decision borders they offer are also relatively simple. Rules referring to prototypes may be a useful alternative in many cases. Selecting the best person for a job requires identification of a "supermen" that may serve as a prototype of a successful candidate. In artificial intelligence case-based reasoning has prominent place and it is well understood that rule-based system are not always the best solution.

Although a lot of effort has been devoted to understanding fuzzy rules prototype-based rules seem to be a new concept. In computational intelligence the *k*-nearest neighbor method uses a reference set of known cases but these cases are too numerous to be useful as a set of rules. In the next section we shall introduce such rules, consider their different form and relation to fuzzy logic rules. In the third section some methods for generating prototype-based rules are described, and the fourth section shows that such rules can provide useful explanation of data in cases when logical rules seem to fail. A short discussion concludes this paper.

Types of rules

Logical rules of several types have been introduced in fuzzy logic [3]. One way to introduce them is by defining predicate functions $P_i(O_j)$ for objects O_j that are evaluated. In crisp logic these predicate functions may simply check if the object has some property, for example if an attribute has some value $A = a_k$ or if the value belongs to some interval. In fuzzy logic predicates are replaced by membership functions, determining the degree to which an object O_j has property P_i. Such membership functions may automatically be created by an iterative procedure in which rules are derived with some initial membership functions, the accuracy of rules is maximized by changing the parameters of the membership functions and the process repeated until convergence. This is done using special "linguistic units" (L-units) in an MLP (multilayer perceptron) network [4] or with an analysis of the nodes created by the Feature Space Mapping (FSM), a constructive neurofuzzy system [5] based on separable transfer functions.

A set of predicate functions applied to the object O gives a feature vector $\mathbf{X} = \{P_1(O), ..P_N(O)\}$. Crisp logical rules (C-rules) based on intervals of the feature values are most comprehensible but they suffer from several drawbacks: 1) only one class is identified as the correct one even when data distributions strongly overlap; 2) reliable crisp rules may not cover all feature space, leaving some vectors unclassified; 3) optimization of the number of errors made by the crisp rule classifiers is difficult because the cost function is discontinues. Crisp rules may be quite misleading, being unstable against small perturbations of input values. A small change in the value of a single feature may lead to a complete change of the predicted class. Interpretation without exploration of alternative diagnoses may in such cases be rather dangerous.

Fuzzy rules (F-rules) do not have these drawbacks but they are not so comprehensible as the crisp rules and they are more complex, involving parameters determining positions and shapes of the membership functions. Fuzzy rules estimate probabilities of different classes but there is a tradeoff between the fuzziness and the degree of precision. If the membership functions are too broad all classes have similar probability. In the opposite case perturbation of the input vector may significantly change classification probabilities, even if the size of the perturbation is within the range of accuracy of the measured input values.

Although various systems differ in their approach to logical rule discovery, their ultimate capability depends on the decision borders they may provide for classification. A very general form of prepositional classification rule is:

$$\text{IF } \mathbf{X} \in K^{(i)} \text{ THEN Class}(\mathbf{X}) = C_i \qquad (1)$$

where C_i =Class($K^{(i)}$), the same as for all vectors in this cluster. In fuzzy logic the operator "belongs to" may have various definitions.

Shapes of clusters used to define such general rules are arbitrary. Techniques for visualization of multidimensional data may provide interactive maps allowing for understanding of the data [6]. If visualization is not used simplifying assumptions regarding the shapes of clusters should be made. How to obtain the smallest number of comprehensible rules, i.e. what is the most appropriate bias?

There is no general answer to such question. For problems with inherent logical structure hyperrectangular decision borders in the feature subspaces are sufficient. Conjunctive rules of the type:

$$\text{IF } \left(X_1 \in X_1 \wedge X_2 \in X_2 \wedge ... X_N \in X_N\right) \text{ THEN Class} = C \quad (2)$$

cover the feature space separating regions from different classes. If X_i are sets of symbolic values, discrete numerical values, or intervals of continuous features, crisp logic rules are obtained. Such rules are generated by many inductive logic covering approaches, decision trees, rough sets [7] and neural rule extraction methods. Hyperrectangular approximation may lead to a large number of rules if complex decision borders are required.

Fuzzy rules with typical triangular or Gaussian membership functions provide decision borders of different shape than those of crisp rules, but still many rules may be required to partition the feature space properly. The crisp form of logical rules is obtained when rectangular membership functions are used. Rectangles allow to define logical linguistic variables for each feature by intervals or sets of nominal values.

M-of-N rules (M out of N antecedents should be true) are sometimes very useful, for example, in medicine "if 2 out of 5 symptoms are present" could be a rather common rule condition. Such rules are natural for threshold logic, but difficult to replace by prepositional rules. Association rules [8] discover simple relations in databases. General form of such rules is:

IF $L(X_1, X_2, ... X_m)$ THAN X_n (p%)

i.e. if some (logical) function $L(\cdot)$ (usually a conjunction) of features $X_i,\ i = 1..m$ is true feature X_n appears in p percent of cases. These rules have the form of prepositional logic although they do not associate conditions with fixed classes but rather take another attribute value as a conclusion. Fuzzy form of such rules may easily be defined replacing the attribute values with their membership functions and the logical function with its fuzzy generalization.

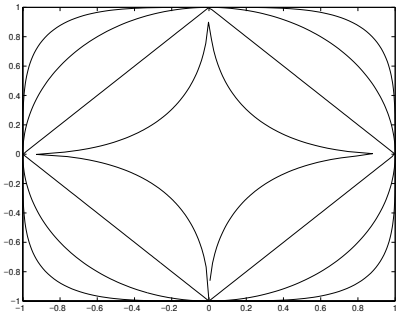

Figure 1: Contours of constant distance for α= 1/2, 1, 2 and 7 (outer contour) exponents of Minkovsky's distance functions.

An alternative way to explain the data is to use a set of prototype-based rules (P-rules):

IF $P = \arg\min_{\mathbf{P}'} D(\mathbf{X},\mathbf{P}')$ THAN Class(**X**) = Class(**P**),

where $D(\mathbf{X},\mathbf{P})$ is a dissimilarity function (usually a distance function). These rules are more general than prepositional rules. In particular the following distance function:

$$D(\mathbf{X},\mathbf{P}) = \max_i W_i|X_i - P_i| \tag{3}$$

has rectangular contours of constant values. If the minimal distance rule is used to find the nearest prototype the decisions borders will have polyhedral shapes. Introducing thresholds d_P rules of the form:
IF $D(\mathbf{X},\mathbf{P}) \leq d_P$ THEN C,
are equivalent to conjunctive crisp rules
IF $X_1 \in [P_1 - d_{P1}/W_1, P_1 - d_{P1}/W_1] \wedge ... \wedge [P_k - d_{Pk}/W_k, P_k - d_{Pk}/W_k]$ THEN C

In contrast to pattern recognition methods such as the k-nearest neighbor method (k-NN), methods that use many reference vectors, the goal here is to find a small number of prototypes and a simple similarity functions that can give understanding of the problem. Similarity function based on Minkovsky's distance is very useful:

$$D(\mathbf{X},\mathbf{P})^\alpha = \sum_{i=1}^{N} W_i|X_i - P_i|^\alpha \tag{4}$$

For large exponents α contours of constant distance become rectangular (Fig. 1).

Detailed relations of similarity functions to membership functions and the S and T-norms in fuzzy logic remain to be investigated. We shall only note that any T-norm, for example a product or a minimum of the membership functions $\mu(X_i - P_i)$ centered at P_i (triangular, Gaussian and other membership functions have additional parameters besides the center), may always be used as a similarity function.

$$S(\mathbf{X},\mathbf{P}) = \prod_{i=1}^{N} \mu(X_i - P_i) \tag{5}$$

Similarity function S may be related to a distance function D by $S = 1/(1+D^2)$. A product of Gaussian membership function gives a multivariate Gaussian $\exp(-||\mathbf{X}-\mathbf{P}||^2)$ centered at **P** with ellipsoidal contours of constant values $||\mathbf{X}-\mathbf{P}||$ =const, i.e. it is equivalent to similarity function obtained from Euclidean distance function. Products of triangular functions give hyperbolic contours. Thus fuzzy rules may be replaced by P-rules with appropriate similarity functions. The reverse does not hold; for example the Manhattan distance function:

$$D(\mathbf{X},\mathbf{P}) = \sum_{i=1}^{N} |X_i - P_i| \tag{6}$$

does not seem to be equivalent to any combination of membership functions and T-norms. Many other distance measures are useful [9], for example Camberra:

$$D_{Ca}(\mathbf{X},\mathbf{Y}) \sum_{i=1}^{N} \frac{|X_i - Y_i|}{|X_i + Y_i|} \tag{7}$$

More general form of rules is obtained if more than one prototype is used in the rule condition: IF among k most similar prototypes P_i class C is the most common than $C(\mathbf{X}) = C$. Such classification rules allow for complex decision borders but may seem more difficult to understand and may require more prototypes (at least k) per class. In approximation problems k-prototype rules will be more useful.

Oblique distribution of data may require linear combination, or non-linear transformation, of input features [9]. The meaning of rules build with such features may be difficult to comprehend. Convex, polyhedral shapes obtained from a union of halfspaces defined by hyperplanes also do not lead to comprehensible rules.

Methods

Prototype-based rules may be created in many ways. Prototypes are useful in forming rules that help to understand the data if: there is a small number of prototypes, only the essential features are used and the most effective distance function is selected. Positions of prototypes should be optimized only if it makes sense, i.e. if interpretation of the P-rules is not lost. Feature selection corresponds to binary weights in the distance function, while discrete or continuos weights may help to increase accuracy, modeling the

process of attention paid to different features during object recognition. Different rules may be used different features and even different similarity measures. Neural similarity-based methods [10] or the Learning Vector Quantization (LVQ) method [11] may be used to generate required prototypes. The LVQ algorithm is constructive and should be extended to include feature selection and selection of the distance function.

In this paper only the simplest approach to create P-rules is presented. It is based on the prototype selection in the nearest neighbor method, preceded by an optimization of distance function and selection of relevant features. Training vectors should be sequentially eliminated from the prototype set until the classification accuracy drops below the assumed target accuracy Δ. Sets of prototype-based rules with increasing accuracy/complexity may be generated taking different values of Δ. The algorithm starts with the training set T and creates the set R containing good prototypes in the following way:

1. Initialization:
   (a) Set the prototype set to the entire training set, R =$T = \{\mathbf{R}_i\}, i = 1..N$.
   (b) Perform the leave-one-out test with the nearest neighbor method on T to find the Δ_1 accuracy.
   (c) Set the target accuracy Δ to Δ_1 and set the lowest accuracy Δ_m that should be considered.
   (d) Define the δ parameter determining steps in which the target accuracy Δ is lowered, for example $\delta = 0.05$.
2. Main loop: until $\Delta < \Delta_m$ do
   (a) For $i = 1$ to N
   (b) Select one vector $\mathbf{R}_i$ from R and set the temporary prototype set to R '=$R - \mathbf{R}_i$.
   (c) Using the leave-one-out test and the current prototype set R ' calculate the prediction accuracy A_c on the whole training set T.
   (d) If $A_c \geq \Delta$ set R =R '.
3. Closing the loop:
   (a) Set $A_e(\Delta) = A_c$ to record the current accuracy.
   (b) Set $R(\Delta)$=R to remember current prototype vectors.
   (c) Change $\Delta \leftarrow \Delta - \delta$.
4. Select the set of prototypes obtained for the highest $A_e(\Delta)$.

Since Δ is set to the leave-one-out accuracy Δ_1 on the entire training set T this algorithm should not degrade the results of the nearest neighbor classifier – in the worst case it will leave all training vectors as prototypes. The threshold value Δ is lowered in several steps δ, allowing for some degradation of the performance as a penalty for reduction of the prototype set. Displaying the function $A_e(\Delta)$ allows to select the optimal value of Δ, depending on the acceptable tradeoff between simplicity/accuracy. The final prototype set should be significantly smaller than the original training set with no or with the minimal degradation of the prediction accuracy.

The Δ parameter controls the final number of P-rules. Since the accuracy is decreased in small steps δ important prototypes that may significantly decrease the accuracy are not removed from the prototype set. If the δ steps are not sufficiently small instead of using cumulative accuracy estimation for the whole selection procedure an additional parameter specifying the allowed threshold of A_c decrease due to the removal of a single prototype may be specified. The selection procedure is repeated for several values of Δ to find a good compromise between classification accuracy and the number of prototype vectors.

If the goal is to maximize performance, estimation of the predicted accuracy $A_e(\Delta)$ should combine results $A_t(\Delta)$ obtained for the rejected vectors, i.e those in the T –R set, and results $A_r(\Delta)$ for the prototype vectors R. Accuracy $A_r(\Delta)$ for the prototype vectors may only be estimated using the leave-one-out test. The goal of the prototype selection algorithm is to obtain a small number of non-redundant prototype cases. However, if all N_r prototypes are non-redundant the accuracy $A_r(\Delta)$ may be quite low. Thus for sufficiently small Δ that corresponds to a small number of prototypes, $A_t(\Delta)$ estimation should be used, while for $\Delta \approx \Delta_1$ the $A_r(\Delta)$ estimation is also important.

The off-line procedure requires an access to all vectors in the training set T in the batch mode. The on-line version of the selection method has also been developed [13].

Computational experiments

Logical rules for a number of datasets were extracted recently [2]. For comparison we have analyzed some of these dataset here.

Iris flowers data, taken from the UCI repository [14], has been used in many previous studies. It contains 3 classes (Iris Setosa, Virginica and Versicolor flowers), 4 attributes (sepal and petal widths and length), 50 cases per class. The Iris dataset is shown in Fig. 2 in two dimensions, x_3 and x_4, that come from feature selection as the most important. In Fig. 3 the reference set obtained by taking the value of $\Delta = 0.96$ from the leave-one-out test on the entire data and

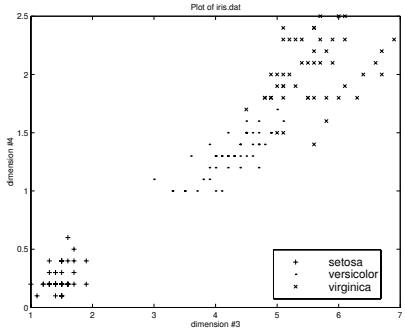

Figure 2: Original 150 Iris data vectors displayed using the last two features.

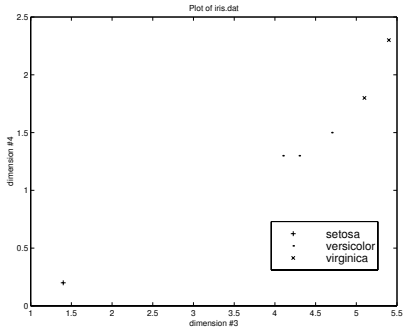

Figure 3: 6 prototype vectors for the Iris data, Euclidean distance.

running the prototype selection algorithm with Euclidean distance functions is displayed. Only 6 prototype vectors remained. A single Iris Setosa prototype is sufficient to perfectly account for this class, 2 prototypes are used for the class virginica, and 3 for the class versicolor. Testing the system on the remaining 144 vectors 5 errors were found, giving accuracy of 97.7%.

In the 10-fold stratified cross-validation test repeated 10 times on the Iris dataset the classical nearest neighbor classifier gives 95.8% accuracy, with 0.3% variance. In the same test P-rules give 95.3% accuracy (an insignificant decrease), with variance 1.7%. On average 6.7 prototypes were found, that is only about 4% of the training set (in 10-fold cross-validation the number of training cases is 135).

Please note (Fig. 3) that the position of the prototypes found by the selection algorithm does not correspond to the most typical objects from the database. They are placed closer to the borders, helping to distinguish cases that are difficult to classify. Thus the selection method works in quite different way than clusterization algorithms.

We have also tried the artificial **3 Monk data** [15] frequently used as a benchmark in machine learning. The Monk problems have inherent logical structure that makes this test appropriate for rule induction methods rather than prototype-based approaches, but it is interesting to see how well P-rules may work here. The *k*-NN method gives good results only if feature selection and/or feature weighting is included [12]. In all cases there are 6 symbolic features, 2 classes and 432 cases in the test set, while the number of training cases varies in each of the problems.

For $k = 1$ on Monk1 (124 training cases) standard *k*-NN gives only 85.9% accuracy using Euclidean distance function. After selection procedure only 10 prototypes were left (2.3% of the entire training set), but the accuracy dropped to 70.6%. This is shown here only to illustrate the importance of the selection of similarity measure (more detailed discussion is in [13]). The best results were obtained with Camberra distance measure (Eq. 7). The feature selection left 3 correct features (head shape, body shape, jacket color) and the leave-one-out test set the Δ target value at 98%. This leads to 15 prototypes giving only 2 errors on the training part (109=124-15) and 94.4% accuracy on the test set. The symbolic nature of data is an additional difficulty here since in 36 cases distances to nearest neighbors from two classes have identical values. In this case additional neighbors may be recruited to make the prediction (24 are correctly resolved in this way). Increasing the Δ to 99.5% adds one prototype (for a total of 16) but makes no errors on the training part and no errors on the test set! This compares rather favorably with the 14 logical formulae obtained from logical rule extraction for this data [2].

For Monk-3 problem (122 training cases) standard *k*-NN with Euclidean distance gives 90.1% accuracy on the test set while selection procedure gives 85.0% accuracy using only 11 prototypes (2.5% of the training set). Using the Camberra distance and performing the feature selection only 2 features have been left. The leave-one-out result on the training set is 93% and the selection procedure left 8 prototypes only. The test set results were much better, with 97.2% correct predictions. Since this problem contains 6 mislabeled vectors in the training set to simulate the effects of noise in the data this is a very good result.

The appendicitis data contains only 106 cases, with 8 attributes (results of medical tests), and 2 classes: 88 cases with acute appendicitis (80.2%) and 18 cases with other problems (19.8%). Very simple classification rules have been found by Weiss and Kapouleas [16] using their PVM (Predictive Value Maximization) approach that makes exhaustive search testing all possible simple rules. The leave-one-out accuracy of their rules is 89.6%, corresponding to only 7 correctly classified additional vectors above the majority rate. So far no classifier was better than that. The purpose of using this data is to check if a simple description of such small and noisy data is possible with similar accuracy. Automatic determination of Δ from the leave-

one-out calculations gave rather low value of 82% and led to 4 prototype vectors only. The leave-one-out result with these 4 prototypes is 88.7%, or just one error more than in case of the best logical rules [2]. This result was obtained without any feature selection and optimization of similarity function.

Hepatobiliary disorders data [17] contain medical records of 536 patients obtained from a university affiliated Tokyo-based hospital, with four types of hepatobiliary disorders. The records include results of 9 biochemical tests and sex of the patient. The same 163 cases as in [17] were used as the test data. For this dataset a large number of crisp logic rules is generated [2]: with 49 rules only about 63% accuracy on the test set was achieved. Although fuzzy rules based on Gaussian or triangular membership functions give higher accuracy (about 75-76%) over 100 are necessary, making the whole set rather incomprehensible. The Δ after leave-one-out was set at 76%. The best similarity functions was of the Camberra type. Feature selection procedure dropped 4 features and the result on the test set was 83.4%. Unfortunately 57 prototypes were selected and P-rules achieved only 64.2% accuracy on the test set. For this data rules in any form do not seem to work, indicating that classes strongly overlap. The best one can do in such cases is to identify the cases that can be reliable classified and assign the remaining cases to pairs of classes [18]. Although this is possible with P-rules we have not tested it yet.

Our final test was on a real medical data, the **melanoma skin cancer**, collected in the Outpatient Center of Dermatology in Rzeszów, Poland [19] (reviewed in details in [20]). Each of the cases belongs to one of the four types of Melanoma: benign, blue, suspicious, or malignant. 13 features include evaluation of asymmetry, border, color and diversity of the skin cancer spots. A linear combination of these features, called a TDS index, has been added as the last feature since it is commonly used in statistical evaluations. The data contains 250 cases, with almost equal distribution of all four classes; additional 26 cases have been provided for testing.

10-fold crossvalidation calculations were performed first on the training set, selecting Manhattan distance function and two features, TDS and C-Blue, as the most important. Perfect accuracy is obtained on the test set (26 cases) and the estimation of accuracy on the whole set was similar to the accuracy $97.4\pm0.3\%$ of our SSV decision tree [2]. The prototype selection procedure left only 13 prototype vectors (7 for the first class and 2 for every other class) still giving 100% accuracy on the test set and 6 errors on the training set (237=250-13 vectors), corresponding to 97.5% accuracy. Reducing the number of prototypes further to 7 leads to a significant decrease in the training set accuracy.

Conclusions

Rule-based classifiers are useful only if rules are reliable, accurate, stable and sufficiently simple to be understood. A new way to understand data using prototype-based rules has been introduced here. It seems to be a useful addition to the traditional ways of data explanation based on crisp or fuzzy logical rules. They may be helpful in cases when logical rules are too complex or difficult to obtain. A small number of prototype-based rules with specific similarity functions associated with each prototype may provide complex decision borders that are hard to approximate using logical systems.

Experiments in cognitive psychology show that human categorization is based on exemplars and prototypes, but not on logical rules defining natural objects in some feature spaces [21]. In the approach presented here similarity functions are used to model the importance of different features in evaluating similarity between the case given and prototypes stored. Prototype-based rules provide an easy way to understand some data. Although similarity measures provide great flexibility in creating various decision borders this may turn to be a disadvantage if our primary goal is to understand the data. Optimized similarity measures may not agree with human intuition. In such cases larger number of prototype examples with simpler similarity measures may be a better solution.

A number of issues requires further investigation. Extension of the simple methods introduced in this paper to adapt prototypes using the learning quantization techniques [11] may be very helpful for small data. Prototypes may also be created by supervised clusterization techniques. An interesting possibility is to use the prototype-based rules to describe exceptions in the crisp or fuzzy logic systems.

Acknowledgments

Support by the KBN, grant 8 T11C 006 19, is gratefully acknowledged.

References

[1] IEEE Transaction on Neural Networks, Special issue on Neural Networks for Data Mining and Knowledge Discovery, Vol. 11, No. 3, May 2000

[2] Duch W, Adamczak R, Grabczewski K, A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks vol. 12, March 2001

[3] Kosko B, Neural Networks and Fuzzy Systems. Prentice Hall 1992

[4] Duch W, Adamczak R, Grąbczewski K, Neural optimization of linguistic variables and membership functions. International Conference on Neural Information Processing

(ICONIP'99), Perth, Australia, Nov. 1999, Vol. II, pp. 616-621

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

[6] Naud A, Duch W, Interactive data exploration using MDS mapping. 5th Conference on Neural Networks and Soft Computing, Zakopane, June 2000, pp. 255-260

[7] Pal S. K, Skowron A, Rough Fuzzy Hybridization A New Trend in Decision-Making. Springer-Verlag (1999)

[8] Agrawal R, Imielinski T, Swami A, Mining association rules between sets of items in large databases. In: ACM Conference on Management of Data 1993, pp. 207-216

[9] Duch W, Jankowski N. New neural transfer functions. Neural Computing Surveys 2 (1999) 639-658

[10] Duch W, Similarity based methods: a general framework for classification, approximation and association. Control and Cybernetics 29, No. 4 (2000)

[11] Kohonen T, Self-Organizing Maps. Springer-Verlag, Heidelberg Berlin, 1995.

[12] Duch W, Grudziński K, Weighting and selection of features. Intelligent Information Systems VII, Ustroń, Poland, 14-18 June 1999, pp. 32-36

[13] Grudziński K., Duch W., SBL-PM: A Simple Algorithm for Selection of Reference Instances for Similarity Based Methods, Intelligent Information Systems (IIS'2000), Physica Verlag, Springer 2000, pp. 99-108

[14] Mertz C.J., Murphy. P.M., UCI repository of machine learning databases, http://www.ics.uci.edu/pub/machine-learning-data-bases.

[15] Thrun S.B. *et al.* (1991) The MONK's problems: a performance comparison of different learning algorithms. Carnegie Mellon University, CMU-CS-91-197

[16] Weiss S.M, Kapouleas I, An empirical comparison of pattern recognition, neural nets and machine learning classification methods. In: J.W. Shavlik and T.G. Dietterich, *Readings in Machine Learning*, Morgan Kauffman Publ, CA 1990

[17] Hayashi Y, Imura A, Yoshida K. 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] Duch W, Adamczak R, Hayashi Y. Neural eliminators and classifiers, 7th International Conference on Neural Information Processing (ICONIP-2000), Dae-jong, Korea, Nov. 2000, ed. by Soo-Young Lee, pp. 1029 - 1034

[19] Bajcar S., Grzegorczyk L. Endangerment by skin cancer among population of south-east part of Poland. Hospital No. 1, Research Report, Rzeszów 1997.

[20] Hippe Z.S. Data mining in medical diagnosis. In: Kącki E. (Ed.) Computers in Medicine, Polish Society of Medical Informatics, Łódź 1999, Vol. 1, pp. 25-34.

[21] Roth I, Bruce V, Perception and Representation, Open University Press, 2nd ed, 1995