A New Method of Improving Classification Accuracy of Decision Tree in Case of Incomplete Samples

Bartosz A. Nowak, Robert K. Nowicki, and Wojciech K. Mleczko

Institute of Computational Intelligence, Czestochowa University of Technology,
Al. Armii Krajowej 36, 42-200 Czestochowa, Poland
{bartosz.nowak,robert.nowicki,wojciech.mleczko}@iisi.pcz.pl

Abstract. In the paper a new method is proposed which improves the classification accuracy of decision trees for samples with missing values. This aim was achieved by adding new nodes to the decision tree. The proposed procedure applies structures and functions of well-known C4.5 algorithm. However, it can be easily adapted to other methods, for forming decision trees. The efficiency of the new algorithm has been confirmed by tests using eleven databases from UCI Repository. The research has been concerned classification but the method is not limited to classification tasks.

Keywords: missing values, C4.5, classification, decision tree.

1 Introduction

In the current stage of the development of computer science, especially computational intelligence, we dispose of many methods designed to data processing and decision making. The important positions are occupied by non-parametric techniques [8,24,25,26], neural networks [1,10,11,23], fuzzy systems [12,22], relational systems [30], classifiers based on Pawlak rough sets [17,19] and decision trees [1,4,27] as well as any hybrid methods [6,7,18,29,31,32]. Actually, all of them have been already adapted to process also incomplete input data. In this area hybrid solutions play important role, especially rough fuzzy systems [15,16] and other high level fuzzy methods [33,34,35]. However, this paper concerns decision trees only.

As in the case of many other decision system, structure of the decision tree is determined by a set of samples applied at design time, i.e. learning set. Each sample concerns single state or observation, and described by a defined set of attributes. Generally, separated samples belong to one or more class, or to none of considered classes. In the paper we assume that every learning sample belongs to exactly one of the considered classes. Moreover, we accept that values of some attributes describing the samples are missing. This subject has been considered by a lot of authors. In many papers there are many various solutions for it. Among others, in [36,37] authors use the internal node strategy in building cost-sensitive decision trees. Another proposition has been formulated in [2]. Authors

L. Rutkowski et al. (Eds.): ICAISC 2013, Part I, LNAI 7894, pp. 448–458, 2013.

use a fuzzy random forest, which is an ensemble of fuzzy decision trees. In a case of missing values on the attribute used in a branch, the sample is further processed by every sub-node with degree of fulfilment divided by number of sub-nodes. A similar solution is proposed in [9], but in application for classification data streams by a fuzzy decision tree. A little more sophisticated approach exist in a popular algorithm C4.5 [20], where degrees of fulfilment are multiplied by factors equal to probabilities of use corresponding sub-nodes.

In the paper authors propose an algorithm that adds alternative nodes to a decision tree in order to improve accuracy of classification in the case when the sample have missing values of attributes. This approach differs from the method in CART [3], where surrogate splits are used, but no new nodes are added to the decision tree. The proposed method applies algorithm C4.5 [20,21], but it is independent from C4.5, therefore authors in the paper have omitted description of C4.5, and the proposed bellow algorithm may also work slightly changed with different methods for building and pruning of decision trees.

The paper is organised as follow. Section 2 contains the genesis of the proposed algorithm, the main idea and details of the method. Section 3 presents the process of experiments and obtained results. It contains also the example of wrapped tree for the simplest benchmark - well known iris classification. The last section is a summary, conclusions and final remarks.

2 Proposed Method

This section presents genesis of proposed algorithm and details of them. We called them WrapTree, because it wrapped the original decision tree by additional branches and nodes which improve the classification accuracy of decision in the case of missing values.

The starting point of the research was the idea of decision tree forest. In this solution many decision trees are created, each for different set of available features. In such ensemble, during classification of a sample only one decision tree is active. It is the tree prepared for work with specific set of attributes compatible with set of available attributes in current sample. If the compatible tree is unavailable, e.g. due to limited system size, the sample is rejected or processed by most appropriate tree using some more sophisticate methods.

During the preliminary studies about forests of trees the following observations have been done:

1. There are many cases when decision trees, which were created with different set of attributes (V_a, V_b) are the same, because they use the same subset of attributes (V_c, $V_c \subseteq V_a \wedge V_c \subseteq V_b$)
2. In some cases decision tree, created with some set of attributes (V_a), contains only one leaf, mainly because of the pruning. In that case there seems to be no reason to create trees with smaller sets of attributes $(V_b \subsetneq V_a)$ than mentioned one-element tree.
3. In the most cases the parts of decision trees close to the root are identical in trees made for various sets of available attributes.

Our goal is to propose the algorithm creating a single tree wrapped by supplementary branches and nodes dedicated to serving the samples with missing features. The reference tree is built for serving complete samples by any known method e.g. C4.5. After wrapping process the resultant tree should assume classification accuracy comparable to mentioned above tree forest. It occurs also in the case of missing features. The total number of branches and nodes should be significantly lower.

Due to mentioned cases, our algorithm does not store identical trees more than once, use original tree as a base for new decision trees, and does not add new nodes for any branches, when it is unnecessary. These features reduce greatly time and resources needed to create the decision tree in comparison to a method, which create decision trees separately for each chosen sets of available attributes.

Because final decision tree is the equivalent to composition of many decision trees, which have been created with different sets of attributes, there is necessity to add proper method checking if chosen attribute has non-missing value.

2.1 Extending of Reference Tree

As was mentioned above, the reference decision tree is prepared by any known algorithm, e.g. C4.5. However, such a tree must be extended to be able to work with supplementary branches and nodes which will be prepared by the proposed algorithm. As a result the reference decision tree will contain four types of branches, i.e.

- Numerical — determine if the value of examined attribute is greater than defined threshold or not. This type of branches occurs only when former branches on the processed path excluded case of inaccessibility of examined attribute value, i.e. the examined in branch attribute has been former examined for the same sample.
- Symbolic — determine if the attribute takes defined value or label or not. As previous one this type of branch occurs if, basing on previous branches, we are sure that value of examined feature is available in the processed sample.
- Numerical or lack — if value of the examined attribute is available they determine if the value of examined attribute is greater than the defined threshold or not. When value of examined attribute is not available the alternative subtree is assigned.
- Symbolic or lack — if value of examined attribute is available they determine if the attribute takes the defined value or label or not. When value of the examined attribute is not available, the alternative subtree is assigned.

Both types of numerical branches have applied in a reference tree shown in Fig. 1.

2.2 Adding of New Nodes

The alternative subtrees, wrapped the reference tree, are created by the proposed procedure (WrapTree) presented below. This procedure is recursive. It adds to

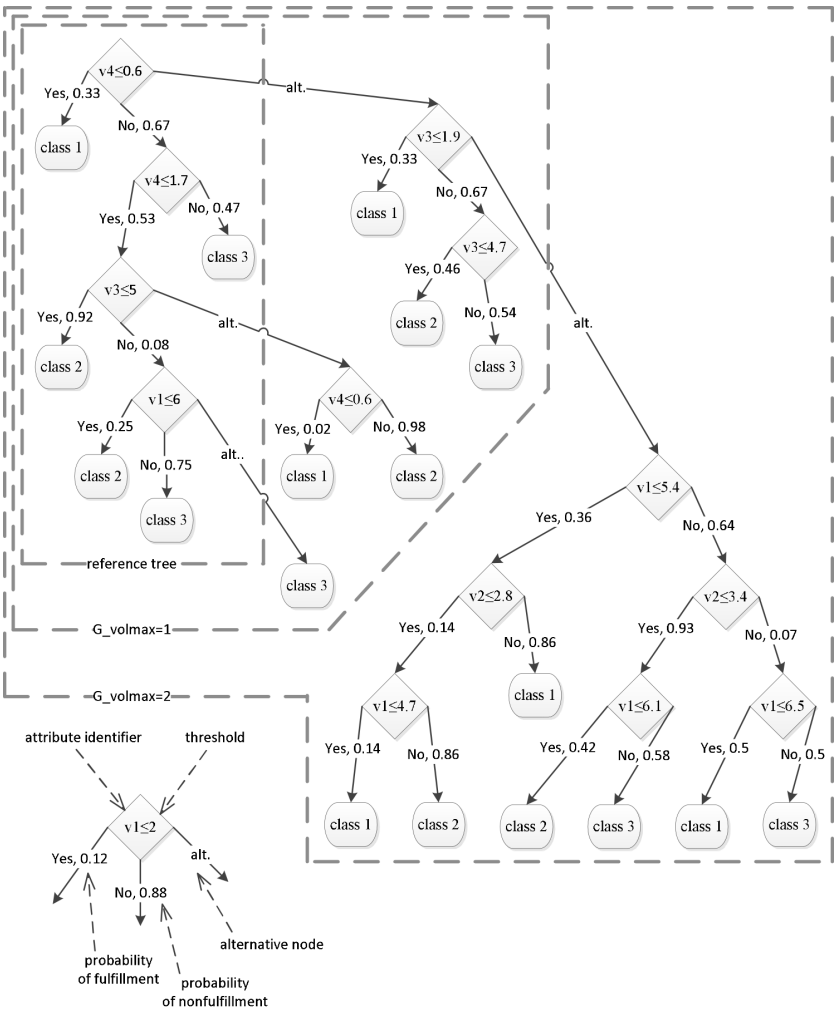

Fig. 1. Results of the algorithm for $G\_volmax = 0, 1, 2$, iris database, 135 samples

some branches of existing decision tree an alternative node. These newly created nodes are used afterwards as a root of a new decision tree, which are created by C4.5, but with limited set of available attributes.

Procedure. WrapTree

Input: a node of decision tree ($tree\_node^{(act)}$), set of already used attributes ($P^{(act)}$), set of excluded attributes ($G^{(act)}$), set of samples ($X^{(act)}$), vector of samples weights ($\mathbf{w}^{(act)}$), maximal size of the $G^{(act)}$ and intensity of a algorithm (constant G_volmax).
Result: Modification of the node $tree\_node^{(act)}$ or its child.

1 **if** $tree\_node^{(act)}$ *is a leaf* **then return**;
2 **if** $||G^{(act)}|| \geq G\_volmax$ **then return**;
3 **if** $\sum_{x_s : x_s \in X^{(act)}} w_s^{(act)} < 1$ **then return**;
4 $v^{(act)}$ = decision attribute of $tree\_node^{(act)}$;
5 **if** $v^{(act)} \notin P^{(act)}$ **then**
  `// new attribute`
6 create an empty node ($tree\_node^{(alt)}$) alternative to $tree\_node^{(act)}$;
7 create a decision sub-tree using $tree\_node^{(alt)}$ as a root, with samples $X^{(act)}$, their weights $\mathbf{w}^{(act)}$ and set of available attributes $V - \left\{G^{(act)} \cup \left\{v^{(act)}\right\}\right\}$;
8 PruneTree($tree\_node^{(alt)}$, $X^{(act)}$, $\mathbf{w}^{(act)}$);
9 WrapTree($tree\_node^{(alt)}$, $P^{(act)}$,$G^{(act)} \cup \{v^{(act)}\}$, $X^{(act)}$, $\mathbf{w}^{(act)}$, G_volmax);
10 $P^{(act)} = P^{(act)} \cup \{v^{(act)}\}$;
11 **foreach** *child node* $tree\_node^{(sub)}$ *of* $tree\_node^{(act)}$ **do**
12 determine $X^{(sub)}, \mathbf{w}^{(sub)}$ for $tree\_node^{(sub)}$; `// in two variants`
13 WrapTree($tree\_node^{(sub)}$, $P^{(act)}$,$G^{(act)}$, $X^{(act)}$, $\mathbf{w}^{(act)}$, G_volmax);

The procedure requires the following input parameters set by operator: constant integer parameter G_volmax greater than 0, but not greater than the number of attributes (n), pointer to node of a decision tree ($tree\_node^{(act)}$), sets of already used ($P^{(act)}$) and excluded ($G^{(act)}$) attributes, set of samples ($X^{(act)}$) and vector of their weights ($\mathbf{w}^{(act)}$). In the first execution of the procedure, root node is pointed, all samples with their initial weights (usually all of them equal to 1) and empty sets of already used and excluded attributes.

Parameter G_volmax defines the intensity of the algorithm, and affects the number of nodes added. We can estimate that the obtained by our procedure wrapped tree is equivalent of an ensemble (forest) of $1 + \binom{n}{1} + \binom{n}{2} ... + \binom{n}{G\_volmax}$ trees, created using various sets of attributes. The first tree of such forest is created using complete samples. The other trees are build using samples with combination of $n - G\_volmax$ to $n - 1$ available attributes. When $G\_volmax = 1$, then proposed algorithm create one decision tree, that is

Table 1. Properties of the used data sets

data set	no. of samples	no. of attributes	no. of classes
dermatology	366	34	6
ecoli	336	7	8
glass	214	9	2
ionosphere	351	34	2
iris	150	4	3
page-blocks	5473	10	5
parkinsons	195	22	2
pendigits	10992	16	10
pima diabetes	768	8	2
vowel	528	10	11
wisconsin	699	9	2

equivalent to a composition of $n+1$ trees ($Tree_a, a = 1 \ldots n+1$). The first one is the reference tree, prepared for complete samples served. The following trees ($Tree_a, a = 2 \ldots n+1$) are created omitting one attribute (v_{a-1}). When $G\_volmax = n$, the algorithm makes tree that is equivalent to composition of decision trees created for every possible combination of available sets, including empty set.

Procedure WrapTree in each execution concerns on the single node of the decision tree and may recursively execute itself on sub-nodes. In the beginning (commands 1-3) WrapTree checks if at least one of stopping condition is fulfilled. These stopping conditions are: verify if the current node is a leaf, and does not have sub-nodes; check if defined maximal level of recursion (G_volmax) has been achieved, verify if a sum of samples weights is too small. After that, the procedure checks if an attribute in the current node has been already used (command 5), if not, then: adds a new empty node ($tree\_node^{(alt)}$), and connects it to current node; create a new sub tree (command 7) using the same set of samples and their weights, but set of attributes reduced by excluded attributes ($G^{(act)}$) and attribute of current node ($v^{(act)}$); then prune that tree (command 8), after that execute WrapTree (command 9) for created and pruned sub-tree with the same settings, but with set of forbidden attributes enlarged by the current attribute ($v^{(act)}$); later adds current attribute to set of already used attributes (command 10). It is worth to mention, that commands 7-8 use C4.5 algorithm, but they can be easily substituted by other decision tree building algorithms. After creation of alternative node procedure processes each sub-node (command 11), which are not the alternative node. At first, procedure determine set of samples and their weights, which should be directed to this sub-node (command 12), and then execute WrapTree (command 13) with determined before set of samples and their weights. In the paper two variants of the method to determine samples and their weights for sub-node (command 12) are proposed, which differ when learning samples have missing values:

1. $X^{(sub)}$ = all samples, that have available current attribute ($v^{(act)}$) and fulfilled condition in $tree\_node^{(act)}$ for child node $tree\_node^{(sub)}$.
$$w_s^{(sub)} = \begin{cases} w_s^{(act)} & \text{if } x_s \in X^{(sub)} \\ 0 & \text{else,} \end{cases}$$
where x_s is a sample with index s.
2. $X^{(sub)}$ = all samples, that have available value for current attribute ($v^{(act)}$) and fulfilled condition in $tree\_node^{(act)}$ for child node $tree\_node_{sub}$; or missing value for current attribute ($v^{(act)}$).
$$w_s^{(sub)} = \begin{cases} w_s^{(cur)} & \text{if } x_s \in X^{(sub)} \wedge v^{(act)} \in P_s \\ w_s^{(cur)} \cdot p^{(cur,sub)} & \text{if } x_s \in X^{(sub)} \wedge v^{(act)} \in G_s \\ 0 & \text{else,} \end{cases}$$
where P_s is a set of attributes with non-missing values for sample x_s, G_s is a set of attributes with missing values for sample x_s, $p^{(cur,sub)}$ is a probability, that learning samples which reached $tree\_node^{(act)}$, and had available values for attribute $v^{(act)}$, were sent to $tree\_node^{(sub)}$. This method is similar to used in C4.5 [20].

By default the first method were used, which has lower accuracy of classification samples in learning set in case of missing values in learning and testing set, but produce smaller decision trees.

Table 2. Efficiency of classification

number of missing values in sample	1		3		
G_volmax	0	1	0	1	3
dataset					
dermatology	.883	**.894**	.863	.892	**.892**
ecoli	.594	**.656**	.465	.486	**.528**
glass	.882	**.882**	.790	.837	**.838**
ionosphere	**.881**	.879	**.871**	.853	.845
iris	.947	**.953**	.767	.693	**.813**
page-blocks	.766	**.825**	.614	.646	**.714**
parkinsons	**.851**	.838	.778	**.838**	**.838**
pendigits	.941	**.959**	.910	.921	**.929**
pima diabetes	**.687**	.681	.628	**.657**	.637
vowel	.703	**.765**	.510	.566	**.605**
wisconsin	.929	**.948**	.946	.943	**.946**
winner	3	8	1	2	9

3 Experiments and Results

All experiments performed to 10-fold cross validation. It states, that whole set of samples is divided into 10 subsets with nearly equal number of samples. All experiments are repeated 10 times and every final result are average for 10 tests.

In each test consecutive subset is chosen as the testing set and remaining 9 subsets as the learning set.

The algorithm was tested using data sets (Table 1) from UCI Repository [14]. Each sample belonged to exactly one class and all attributes were numerical. In the paper efficiency of classification is computed not as simple accuracy of classification but as average accuracy of classification for samples in each class,

$$efficiency = \frac{1}{m} \sum_{j=1...m} \left(\frac{1}{||\omega_j||} \sum_{x_s : x_s \in \omega_j} correctly\_classified(x_s) \right), \qquad (1)$$

where m is a number of samples, ω_j is j class, $||\omega_j||$ is number of testing samples that belongs to ω_j, $correctly\_classified(x_s)$ is a logical function, which states if sample x_s was properly classified.

For testing purposes databases were prepared in two variants (a number of missing values in sample = {1,3}), according to number of attributes with missing values for each sample. Distribution of missing values within data sets was pseudo-random, but constructed system tried to enforce the same number of missing values for each attribute.

Table 3. Number of nodes in decision tree

number of missing values in sample	1		3		
G_volmax	0	1	0	1	3
dataset					
dermatology	32.2	179.0	54.8	239.4	2236.7
ecoli	57.4	216.1	76.8	207.5	664.2
glass	14.0	53.7	19.8	60.4	204.2
ionosphere	27.2	153.1	27.4	143.0	1294.1
iris	21.2	44.5	39.8	63.5	100.8
page-blocks	128.8	627.7	131.2	534.4	2914.1
parkinsons	22.6	89.0	20.4	79.8	457.0
pendigits	908.0	5737.6	1796.8	7782.7	61448.3
pima diabetes	24.0	88.8	12.8	50.4	271.3
vowel	190.0	875.0	302.0	994.0	4695.0
wisconsin	33.2	131.5	51.8	167.1	563.9
average difference	–	+338,9%	–	+261,0%	+2045,3%

Experiments were performed with standard for C4.5 parameters of building tree and pruning. Parameter G_volmax was set to values: 0, which means that algorithm WrapTree was disabled; 1 and 3. Parameter $G\_volmax = 3$ was tested only for samples with 3 missing parameters, because all decision tree created with G_volmax greater than number of missing values in testing sample works exactly the same.

Table 4. Number of the used nodes during classification

number of missing values in sample	1		3		
G_volmax	0	1	0	1	3
dataset					
dermatology	5.4	5.2	7.3	6.0	5.9
ecoli	8.3	6.0	17.6	10.3	6.3
glass	4.0	3.8	7.6	5.6	4.8
ionosphere	5.6	5.4	6.6	6.0	5.9
iris	6.4	4.3	23.3	10.2	4.9
page-blocks	11.8	8.6	19.1	11.0	7.8
parkinsons	4.6	4.3	5.1	4.3	4.2
pendigits	13.2	10.3	28.7	15.5	11.0
pima diabetes	5.3	4.4	5.6	5.0	4.4
vowel	11.8	8.3	29.7	14.2	9.0
wisconsin	5.3	4.4	11.1	7.0	5.4
average difference	–	-17.5%	–	-32.3%	-44.6%

Results of the experiments are shown in tables: Table 2 – average efficiency of classification of each class according to 1, Table 3 – mean number of nodes in decision tree, Table 4 – average number of nodes used during classification of testing samples.

4 Final Remarks

In the paper, authors presented a new method for improving accuracy of classification by decision trees in case of samples with missing values. The effectiveness of a solution has been confirmed by series computer simulations. As expected, the tests showed that size of decision trees significantly increased after proposed procedure execution. The size is notwithstanding smaller than corresponding tree forest. Moreover, the wrapped tree use smaller number of nodes than other solution designed to process samples with missing features.

The future works with presented idea will concerns extending the interpretability of the knowledge contained in the tree. In this subject the inspiration could be a proposition that comes from fuzzy systems [5]. Also the ensembles of wrapped trees combined with other types of classification [28] could be promising. They could use e.g. AdaBoost or bagging metaalgorithms [13].

References

1. Bartczuk, Ł., Rutkowska, D.: Type-2 fuzzy decision trees. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 197–206. Springer, Heidelberg (2008)
2. Bonissone, P., Cadenas, J.M., Carmen Garrido, M., Andrés Díaz-Valladares, R.: A fuzzy random forest (2010)

3. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth Int. Group (1984)
4. Brodley, C.E., Utgoff, P.E.: Multivariate decision trees (1995)
5. Cpalka, K.: A method for designing flexible neuro-fuzzy systems. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 212–219. Springer, Heidelberg (2006)
6. Cpałka, K.: On evolutionary designing and learning of flexible neuro-fuzzy structures for nonlinear classification. Nonlinear Analysis: Theory, Methods & Applications 71(12), 1659–1672 (2009)
7. Gabryel, M., Scherer, R.: Determining fuzzy relation by evolutionary learning in neuro-fuzzy systems. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J. (eds.) Computational Intelligence: Methods and Applications, pp. 176–182. Academic Publishing House EXIT (2008)
8. Greblicki, W., Rutkowski, L.: Density-free bayes risk consistency of nonparametric pattern recognition procedures. Proceedings of the IEEE 69(4), 482–483 (1981)
9. Hashemi, S., Yang, Y.: Flexible decision tree for data stream classification in the presence of concept change, noise and missing values. Data Mining and Knowledge Discovery 19, 95–131 (2009)
10. Haykin, S., Network, N.: A comprehensive foundation. Neural Networks 2 (2004)
11. Horzyk, A., Tadeusiewicz, R.: Self-optimizing neural networks. In: Yin, F.-L., Wang, J., Guo, C. (eds.) ISNN 2004. LNCS, vol. 3173, pp. 150–155. Springer, Heidelberg (2004)
12. Korytkowski, M., Rutkowski, L., Scherer, R.: From ensemble of fuzzy classifiers to single fuzzy rule base classifier. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 265–272. Springer, Heidelberg (2008)
13. Korytkowski, M., Scherer, R., Rutkowski, L.: On combining backpropagation with boosting. In: 2006 International Joint Conference on Neural Networks, IEEE World Congress on Computational Intelligence, Vancouver, BC, Canada, pp. 1274–1277 (2006)
14. Mertz, C.J., Murphy, P.M.: UCI machine learning repository, `http://archive.ics.uci.edu/ml/datasets.html`
15. Nowicki, R.: On combining neuro–fuzzy architectures with the rough set theory to solve classification problems with incomplete data. IEEE Trans. on Knowledge and Data Engineering 20(9), 1239–1253 (2008)
16. Nowicki, R.: Rough–neuro–fuzzy structures for classification with missing data. IEEE Trans. on Systems, Man, and Cybernetics—Part B: Cybernetics 39(6), 1334–1347 (2009)
17. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences 11(5), 341–356 (1982)
18. Przybył, A., Cpałka, K.: A new method to construct of interpretable models of dynamic systems. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part II. LNCS, vol. 7268, pp. 697–705. Springer, Heidelberg (2012)
19. Qian, Y., Dang, C., Liang, J., Zhang, H., Ma, J.: On the evaluation of the decision performance of an incomplete decision table. Data & Knowledge Engineering 65(3), 373–400 (2008)
20. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
21. Quinlan, J.R.: Improved use of continuous attributes in c4.5. Journal of Artificial Intelligence Research 4, 77–90 (1996)

22. Rutkowska, D., Nowicki, R.: Implication-based neuro–fuzzy architectures. International Journal of Applied Mathematics and Computer Science 10(4), 675–701 (2000)
23. Rutkowska, D., Rutkowski, L., Nowicki, R.: On processing of noisy data by fuzzy inference neural networks. In: Proceedings of the IASTED International Conference, Signal and Image Processing, Nassau, Bahamas, pp. 314–318 (October 1999)
24. Rutkowski, L.: Sequential estimates of probability densities by orthogonal series and their application in pattern classification. IEEE Transactions on Systems, Man and Cybernetics SMC-10(12), 918–920 (1980)
25. Rutkowski, L.: Adaptive probabilistic neural networks for pattern classification in time-varying environment. IEEE Transactions on Neural Networks 15(4), 811–827 (2004)
26. Rutkowski, L.: Generalized regression neural networks in time-varying environment. IEEE Transactions on Neural Networks 15(3), 576–596 (2004)
27. Rutkowski, L., Pietruczuk, L., Duda, P., Jaworski, M.: Decision trees for mining data streams based on the McDiarmid's bound. IEEE Transactions on Knowledge and Data Engineering 25 (2013)
28. Scherer, R.: Boosting ensemble of relational neuro-fuzzy systems. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 306–313. Springer, Heidelberg (2006)
29. Scherer, R., Korytkowski, M., Nowicki, R., Rutkowski, L.: Modular rough neuro-fuzzy systems for classification. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 540–548. Springer, Heidelberg (2008)
30. Scherer, R., Rutkowski, L.: A fuzzy relational system with linguistic antecedent certainty factors. In: Rutkowski, L., Kacprzyk, J. (eds.) Proceedings of the Sixth International Conference on Neural Network and Soft Computing. Advances in Soft Computing, pp. 563–569. Springer, Heidelberg (2003)
31. Scherer, R., Rutkowski, L.: Neuro-fuzzy relational classifiers. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 376–380. Springer, Heidelberg (2004)
32. Scherer, R., Rutkowski, L.: Connectionist fuzzy relational systems. In: Hagamuge, S.K., Wang, L. (eds.) Computational Intelligence for Modelling and Prediction. SCI, vol. 2, pp. 35–47. Springer, Heidelberg (2005)
33. Starczewski, J.T.: On defuzzification of interval type-2 fuzzy sets. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 333–340. Springer, Heidelberg (2008)
34. Starczewski, J.T.: A type-1 approximation of interval type-2 FLS. In: Di Gesù, V., Pal, S.K., Petrosino, A. (eds.) WILF 2009. LNCS, vol. 5571, pp. 287–294. Springer, Heidelberg (2009)
35. Starczewski, J.T.: General type-2 fls with uncertainty generated by fuzzy rough sets. In: FUZZ-IEEE, pp. 1–6 (2010)
36. Zhang, S.: Decision tree classifiers sensitive to heterogeneous costs. Journal of Systems and Software 85(4), 771–779 (2012)
37. Zhang, S., Qin, Z., Ling, C., Sheng, S.: "Missing is useful": missing values in cost-sensitive decision trees. IEEE Transactions on Knowledge and Data Engineering 17(12), 1689–1693 (2005)