Electronic Notes in Theoretical Computer Science 82 No. 4 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume82.html 11 pages

Hybrid Classifier Based on Rough Sets and Neural Networks

Jarosław Stepaniuk and Katarzyna Kierzkowska [1]

Department of Computer Science
Białystok University of Technology
Wiejska 45A, 15-351 Białystok, Poland

Keywords: rough sets, neural networks, decision rules

Abstract

In this paper the methods of objects classification based on rough set theory and artificial neural networks are presented. The results of the experiments based on a hybrid classifier using decision rules and neural network are discussed.

1 Introduction

A *classification algorithm* is an algorithm, which permits us to repeatedly make a forecast in new situations on the basis of accumulated knowledge. In this paper we consider a classification related to construction of a classifying algorithm which on the basis of current knowledge will be applied to classify objects previously unseen. Each new object will be assigned to a class belonging to a predefined set of classes on the basis of observed values of suitably chosen attributes.

Many approaches have been proposed for constructing classification algorithms, among them we would like to point out statistical techniques, neural networks, decision trees and decision rules [2], [3].

The popular method for classification algorithm construction is based on learning rules from examples. One can use rough set methods to discover rules from data sets. The methods based on calculation of reducts allow to compute, for given data, the descriptions of concepts by means of decision rules (see [5]).

In the paper we discuss a method of treating decision rules as a source for new attributes. Using those rules we construct new set of data that is training set for artificial neural network. This approach is inspired by [10], [11].

[1] Email: jstepan@ii.pb.bialystok.pl

The paper is organized as follows. In Section 2 we introduce basic notions. In Section 3 we discuss decision rules as a source for new set of data, which is used as training set for neural network. In Section 4 the results of experiments on three data sets are included.

2 Basic notions

In this section we recall voting strategies for new object classification. We also formulate the problem of conflict between decision rules.

Standard voting use decision rules to classify the objects. The rules fire if their antecedent is not in conflict with the present object. In the election process among the firing rules, each rule gets a certain number of votes in favor of the decision value it indicates. The greatest number of votes indicates the decision.

A decision table is a tuple $DT = (U, A \cup \{d\})$, where U is a set of objects, A is a set of attributes and d is the decision attribute. Let $Supp_{DT}(r)$ be the number of objects for which attribute values satisfy the premise of the decision rule r and the decision given by the rule r is the same as in the decision table. Let $Match_{DT}(r)$ be the set of objects which attribute values satisfy the premise of the rule r.

We recall Support/Firing Weight method of weights computation, which we use for classifying objects. We compare the results of voting based on Support/Firing Weight method with the hybrid classifier.

The Support/Firing Weight method use support of rule. The vote of every rule from the set of rules is the number of objects that support this rule. This weight represents a percent of votes that is given for every decision class with relation to all the rules that recognize a testing object.

When we have the set of decision rules the classification of a new object relies on finding the rules with premises satisfied for the object. These rules may have different decisions. This situation is called the conflict between rules (see Figure 1). There are some methods that resolve these conflicts, for example the classification with hybrid classifier.

3 Hybrid Classifier

The idea of using artificial neural networks to resolve conflicts between rules is proposed in [10], [11]. Finding the best strategy in the specific situation requires the experiments connected with the selection of methods for conflict resolution. For example in the Standard Voting we must choose the weight in advance. In case of using neural networks the weights are calculated. When we use simply neural network without hidden layers, we can interpret the weights. They represent how the rules influence on the decision.

We can split our system into two parts: the transformation of the decision table and the application of neural networks (see Figure 2).

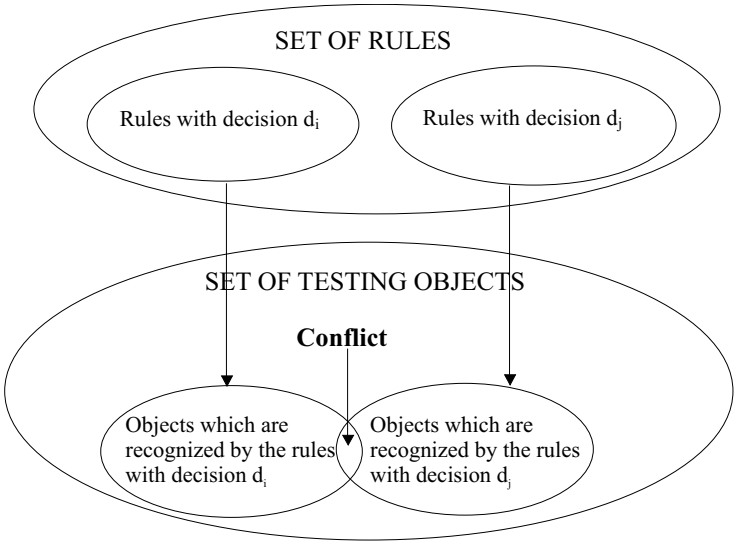

Fig. 1. Conflict Between Rules

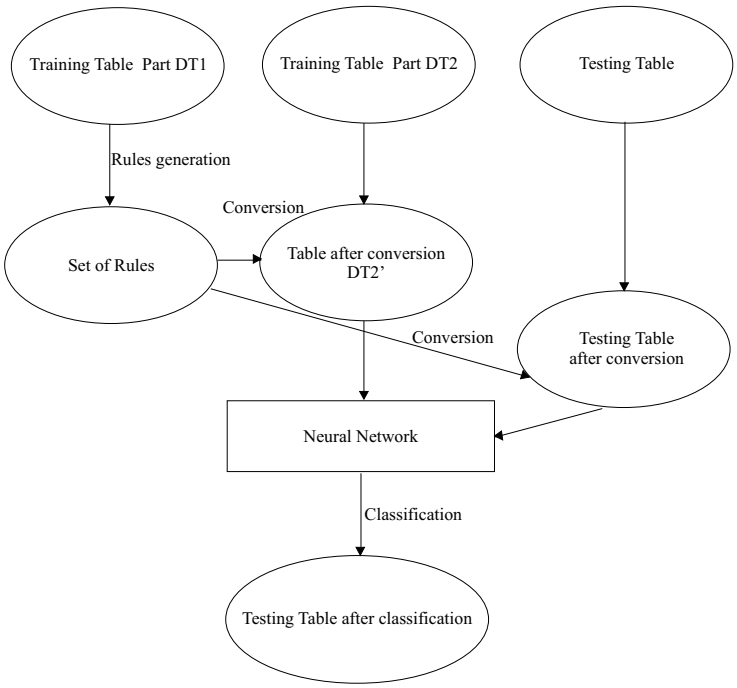

Fig. 2. General Scheme of Our System

3.1 The transformation of the training table

Assume a decision table $DT = (U, A \cup \{d\})$, called training table is given, where U is the set of objects, A is a set of attributes and d is the decision. The training table is split into two parts $DT_1 = (U_1, A \cup \{d\})$ and $DT_2 = (U_2, A \cup \{d\})$, where $U_1 \cup U_2 = U$ and $U_1 \cap U_2 = \emptyset$. From table $(U_1, A \cup \{d\})$ we generate the rules and obtain *Rule_Set*.

The discussed approach is implemented in ConRes (**Con**flict **Res**olution). The ConRes program transforms the table $DT_2 = (U_2, A \cup \{d\})$ to a new table $DT_2' = (U_2, A_{Rule\_Set} \cup \{d\})$, where A_{Rule_Set} is the set of attributes defined by means of rules, the values of the attributes are calculated during the transformation.

For every rule r of the form $if\ \lambda\ then\ d = v$ the attribute value of $a_r(u)$ is calculated using the type of transformation and the object u (see Figure 2).

3.1.1 Simple transformation

The values of the new attribute a_r for every training object u and the rule r are defined as follows:

$a_r(u) = 1$ – if the attribute values of given object satisfy the premise of the rule r and the decision given by the rule r is the same as in the decision table.

$a_r(u) = 0$ – if the object u is not recognized by the rule r

$a_r(u) = -1$ – if the object u is recognized by the rule r, but the decision given by the rule is different from the decision in the table.

For a testing object u the attribute values are calculated as follows:

$a_r(u) = 1$ – if the attribute values of the given object u satisfy the premise of the rule r.

$a_r(u) = 0$ – if the attribute values of the given object u does not satisfy the premise of the rule r.

3.1.2 Partial transformation

We assume that the premise λ of the rule r is composed of several selectors. Therefore, the rule r is represented as follows:

$$if\ (\lambda_1\ and\ \ldots\ and\ \lambda_{M_r})\ then\ d = v,$$

where M_r is the number of attributes in the rule r. Let u be a training or testing object and let $L_r(u)$ be the number of attributes that have the same value as the value of attributes on object u.

We define ω_r for training object u as follows:

$$\omega_r(u) = \begin{cases} 1 & \text{if} \quad d(u) = v \\ -1 & \text{otherwise} \end{cases}.$$

For the testing object u we define $\omega_r(u) = 1$.

The value of attribute a_r is defined by

$a_r(u) = \omega_r(u) \cdot \frac{L_r(u)}{M_r}$

The values of the new attribute for every object and for given rule depend on the number of attributes in the premise of the rule having the same value in the rule and on the object.

The a_r values are ranged from -1 to 1 for the training set, and from 0 to 1 for the testing set. They are representing the degree of matching of the given

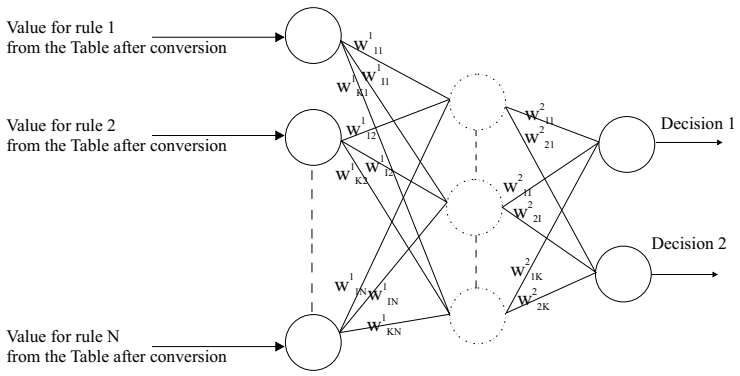

Fig. 3. Construction of Neural Network

rule by the given object.

3.2 Construction of neural network

In input layer the number of neurons is equal to the number of rules + 1 (corresponding to bias). There are as many neurons as decision classes in output layer. In experiments we use neural network with and without hidden layer. General scheme of the neural network construction is depicted in Figure 3. The training table for constructed neural network is $DT_2' = (U_2, A_{Rule\_Set} \cup \{d\})$.

4 Experiments

In this section we present the results of classification with program ConRes. We use three sets of objects "Iris", "Diabetes" and "Australian". Before classification we use Rosetta system to split the sets for two parts each, and we make discretization using Boolean reasoning algorithm. From one subset we generate rules and this subset is used to learning neural network. The second subset is used to test the classifier.

For training set we use two methods of transformation:

- -1, 0, 1 transformation, when the sign of the new attribute depends on decision
- 0, 1 transformation, the same as for testing set (decision does not have influence on sign)

4.1 Iris data

The number of objects in Iris database is equal to 150 (for more details see e.g. [3]). We split this set into two parts, 75 elements each. From first subset, after discretization, we generate 8 rules, and this set is used to train the neural network. The second subset, after discretization, is used for testing

the classifier. In Table 1 we present the results of classification with double cross-validation.

Program	Method	Transformation	Training %	Testing %
Rosetta	SV	-	88	79
Neural network without hidden layer				
Rosetta+ConRes	0,1, P	simple	85	61
Rosetta+ConRes	0,1, P	partial	55	55
Rosetta+ConRes	0,1, BP	simple	97	95
Rosetta+ConRes	0,1, BP	partial	98	94
Rosetta+ConRes	-1,0,1, P	simple	63	65
Rosetta+ConRes	-1,0,1, P	partial	65	37
Rosetta+ConRes	-1,0,1, BP	simple	99	95
Rosetta+ConRes	-1,0,1, BP	partial	100	80
Neural Network	P	-	37	36
Neural Network	BP	-	38	42
Neural network with one hidden layer				
Rosetta+ConRes	0,1	simple	97	96
Rosetta+ConRes	0,1	partial	100	94
Rosetta+ConRes	-1,0,1	simple	97	96
Rosetta+ConRes	-1,0,1	partial	97	96
Neural Network	BP	-	43	38

Table 1
Results of Experiments for Iris Data Set (SV-Standard Voting, P-Perceptron, BP-Back Propagation)

When we use back propagation (BP) method for learning neural network the results of classification are better than in the case of learning based on perceptron rule. Simple transformation is better than partial. When we use partial transformation neural network is adapted to training set and that is why classification for testing set is worse. Transformation 0, 1 give better results than transformation -1, 0, 1. For testing set we always use transformation 0, 1, and that is why the results are better when we use the same transformation for training sets. The results are worse when we use only neural network classifier.

4.2 Australian data

This data set concerns credit card applications [3]. All attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. We split this set into two parts, 345 elements each. From first subset we generate 17 rules and it is used to train the neural network. In Table 2 we present the results of classification with double cross-validation.

Program	Method	Transformation	Training %	Testing %
Rosetta	SV	-	15	7
Neural network without hidden layer				
Rosetta+ConRes	0,1, P	simple	52	51
Rosetta+ConRes	0,1, P	partial	59	62
Rosetta+ConRes	0,1, BP	simple	62	59
Rosetta+ConRes	0,1, BP	partial	84	79
Rosetta+ConRes	-1,0,1, P	simple	52	51
Rosetta+ConRes	-1,0,1, P	partial	78	52
Rosetta+ConRes	-1,0,1, BP	simple	62	59
Rosetta+ConRes	-1,0,1, BP	partial	100	76
Neural Network	P	-	51	52
Neural Network	BP	-	47	47
Neural network with one hidden layer				
Rosetta+ConRes	0,1	simple	63	63
Rosetta+ConRes	0,1	partial	100	81
Rosetta+ConRes	-1,0,1	simple	63	63
Rosetta+ConRes	-1,0,1	partial	92	82
Neural Network	BP	-	51	49

Table 2
Results of Experiments for Australian Data Set (SV-Standard Voting, P-Perceptron, BP-Back Propagation)

The most effective method is back propagation in this case. When we use simple transformation the results are worse than when we use partial transformation. It can arise from resemblance between objects in training set and in testing set. The worst method is standard voting in this case, because

we have small number of rules.

4.3 Diabetes data

This set consists of 107 children data which are suffering with diabetes mellitus (for more details see [8], [9]).

4.3.1 Experiment 1

We split this set in two parts, 54 and 53 elements. From first subset after discretization we generate 15 rules. In Table 3 we present the results of classification with double cross-validation.

Program	Method	Transformation	Training %	Testing %
Rosetta	SV	-	72	11
Neural network without hidden layer				
Rosetta+ConRes	0,1, P	simple	52	53
Rosetta+ConRes	0,1, P	partial	57	58
Rosetta+ConRes	0,1, BP	simple	74	53
Rosetta+ConRes	0,1, BP	partial	80	66
Rosetta+ConRes	-1,0,1, P	simple	52	53
Rosetta+ConRes	-1,0,1, P	partial	91	60
Rosetta+ConRes	-1,0,1, BP	simple	74	53
Rosetta+ConRes	-1,0,1, BP	partial	100	53
Neural Network	P	-	56	55
Neural Network	BP	-	54	52
Neural network with one hidden layer				
Rosetta+ConRes	0,1	simple	74	49
Rosetta+ConRes	0,1	partial	92	64
Rosetta+ConRes	-1,0,1	simple	74	49
Rosetta+ConRes	-1,0,1	partial	100	69
Neural Network	BP	-	52	49

Table 3
Results of Experiment 1 for Diabetes Data Set (SV-Standard Voting, P-Perceptron, BP-Back Propagation)

The best results have been obtained for 0, 1 partial transformation and back propagation. For this set classification with partial transformation is better than with simple transformation. The worst method is standard voting (the most of objects are not recognized).

4.3.2 Experiment 2

We split the set in three parts, 18, 36, and 53 elements. We generate 5 rules from first subset. The second subset we use for training the neural network. In Table 4 we present the results of classification with double cross-validation.

Program	Method	Transformation	Training %	Testing %
Rosetta	SV	-	72	43
Neural network without hidden layer				
Rosetta+ConRes	0,1, P	simple	53	53
Rosetta+ConRes	0,1, P	partial	53	53
Rosetta+ConRes	0,1, BP	simple	69	43
Rosetta+ConRes	0,1, BP	partial	69	62
Rosetta+ConRes	-1,0,1, P	simple	79	62
Rosetta+ConRes	-1,0,1, P	partial	51	49
Rosetta+ConRes	-1,0,1, BP	simple	91	57
Rosetta+ConRes	-1,0,1, BP	partial	100	45
Neural Network	P	-	57	54
Neural Network	BP	-	47	53
Neural network with one hidden layer				
Rosetta+ConRes	0,1	simple	66	47
Rosetta+ConRes	0,1	partial	69	66
Rosetta+ConRes	-1,0,1	simple	91	66
Rosetta+ConRes	-1,0,1	partial	100	62
Neural Network	BP	-	57	56

Table 4
Results of Experiment 2 for Diabetes Data Set (SV-Standard Voting, P-Perceptron, BP-Back Propagation)

The best results have been obtained when we use 0, 1 partial transformation and back propagation for neural network with one hidden layer.

Conclusions

Transformation 0, 1 is better the transformation -1, 0, 1. For testing set we always use 0, 1 transformation, because we do not know the decision. When we use the same transformation for training and testing sets the results are better. Classification is more effective for training set, when we use -1, 0, 1 transformation, but then the neural network is more compatible for this set and classification for testing set is worse. Partial transformation is better than simple transformation. It does not make difference if we split sets in two or in three parts. Simple neural network for conflict resolution allows for easy interpretation for weights. The weights show how the given rule is important for classification. When we use the neural network for conflict resolution with one hidden layer the results are better, but in this case we can not simply interpret the weights. Hybrid classifier gives better results than neural network oneself.

Acknowledgements

We would like to thank Professor Andrzej Skowron for valuable comments and critical remarks.

References

[1] Kościuk K., Stepaniuk J.: Conflict Resolution Between Rules, Zeszyty Naukowe Politechniki Białostockiej Informatyka nr 1, J. Stepaniuk (Ed.), 2002, 97–118.

[2] Mitchell T.M.: Machine Learning. Mc Graw–Hill, Portland, 1997.

[3] Michie D., Spiegelhalter D., Taylor C.C.: Machine Learning, Neural and Statistical Classification, Ellis Horwood Limited, England, 1994.

[4] Pal S.K., Skowron A. (Eds.): Rough–Fuzzy Hybridization: A New Trend in Decision Making. Springer–Verlag, Singapore, 1999.

[5] Pawlak Z.: Rough Sets. Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, Dordrecht, 1991.

[6] Polkowski L., Skowron A. (Eds.): Rough Sets in Knowledge Discovery 1 and 2, Physica–Verlag, Heidelberg, 1998.

[7] Rosetta: Home Page, http://www.idi.ntnu.no/~aleks/rosetta/.

[8] Stepaniuk J.: Rough Set Data Mining of Diabetes Data. Proceedings of the Eleventh International Symposium on Methodologies for Intelligent Systems, Foundations of Intelligent Systems (ISMIS'99), June 8–11 1999, Warsaw, Lecture Notes in Artificial Intelligence 1609, Springer–Verlag, Berlin, 457–465.

[9] Stepaniuk J.: Knowledge discovery by application of rough set models, L. Polkowski, S. Tsumoto, T.Y. Lin, (Eds.), Rough Sets: New Developments, Physica–Verlag, Heidelberg, 2000, 137–233.

[10] Szczuka M. S.: Symbolic methods and artificial neural networks in classifier construction, Ph. D. thesis, supervisor A. Skowron, Warsaw University, 1999.

[11] Szczuka M. S.: Refining Classifiers with Neural Networks, International Journal of Intelligent Systems, Vol. 16 No 1, 2001, 39–55.