Harnessing classifier networks – towards hierarchical concept construction

Dominik Ślęzak[1,2], Marcin Szczuka[3], and Jakub Wróblewski[2]

[1] Department of Computer Science, University of Regina
Regina, SK, S4S 0A2, Canada
[2] Polish-Japanese Institute of Information Technology
Koszykowa 86, 02-008 Warsaw, Poland
[3] Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
{slezak,jakubw}@pjwstk.edu.pl, szczuka@mimuw.edu.pl

Abstract. The process of construction and tuning of classifier networks is discussed. The idea of relating the basic inputs with the target classification concepts via the internal layers of intermediate concepts is explored. Intuitions and relationships to other approaches, as well as the illustrative examples are provided.

1 Introduction

If we take a look at the standard approach to classification and decision support with use of learning systems, we quickly realize that it does not always fit the purpose. Equipped with the hypothesis formation (learning) algorithm, we attempt to find a possibly direct mapping from the input values to decisions. Such an approach does not always result in success, because of various reasons. We address the situation when the desired solution should be more fine-grained, namely, it should have an internal structure. Although possibly hard to find and learn, such architecture repays us by providing significant extensions in terms of flexibility, generality and expressiveness of the yielded model.

We attempt to show our view on the process of construction and tuning of hierarchical structures of concepts. We address these structures as *classifier networks*, where the input concepts correspond to the classified objects or their behavior with respect to the standard classifiers and the output (target) concept reflects the final classification. The basic idea is that the relationship between such input and output concepts is not direct but based on the internal layers of intermediate concepts, which help in more reliable transition from the basic information to possibly compound classification goal. We strive against formalization of this approach with use of analogies rooted in other areas, such as artificial neural networks [2, 3], ensembles of classifiers [1, 7, 12], and layered learning [11]. Moreover, in parallel to introduction of the formalism that describes our classifier networks, we make an effort to illustrate them with examples of actual models as described in our earlier, application-oriented papers [9, 10].

The paper starts with general overview sketching main points of the proposed approach. We provide some intuitions and familiarize the reader with mechanisms present in our proposed model. Then, we go step-by-step through formalization describing the kinds of dependencies that drive the whole approach. Where possible, we provide examples to better ground the ideas.

2 Hierarchical learning and classification

Let us start by explaining how we intend to treat the notion of a concept. In general, a concept is an element drawn from a parameterized concept space. By a proper setting of these parameters we choose the right concept. Note, that we do not initially demand that all concepts come from the same space.

In our approach, a concept represents a basic element acting on the basis of information originating from other concepts or directly from the data source. To better depict the whole structure, it is convenient to exploit the analogy with artificial neural networks. In this case, a concept corresponds to a signal transmitted through a neuron – the basic computing unit. Dependencies between concepts, their precedence and importance, are represented by weighted connections between nodes. Similarly to the feedforward neural network, operations can be performed from bottom to top. They can correspond to the following goals:

Construction of compound concepts from the elementary ones. It can be observed in case-based reasoning [4], layered learning [11], as well as rough mereology [6] and rough neurocomputing [5], where we want to approximate target concepts step by step, using the simpler concepts that are easier to learn directly from data. This corresponds to the generalization of simple concepts.

Construction of simple concepts from the advanced ones. It can be considered for the synthesis of classifiers, where we start from compound concepts reflecting the behavior of a given object with respect to particular, often compound classification systems, and we tend to obtain a very simple concept of a decision class where that object should belong to. This corresponds to the instantiation of general concept in a simpler, more specialized concept.

Obviously, we do not assume that the above are the only possible types of constructions. For instance, in a classification problem, decision classes can have a compound semantics requiring gradual specification corresponding to the first type of construction. Then, once we reach an appropriate level of expressiveness, we follow the second above type to synthesize those compound specifications towards obtaining the final response of the classifier network.

3 General network architecture

When considering hierarchical structures for compound concept formation, several issues pop-up. At the very general level of hierarchy construction/learning, one has to make choices with respect to homogeneity and synchronization. We mention below how these factors determine the complexity of construction task.

Homogeneous vs. heterogeneous. At each level of hierarchy we make choice of the concept space to be used. In the simplest case each node implements the same type of mapping. The nodes differ only by choice of parameters. We have studied a system of this kind (i.e. fully homogeneous) in [9, 10] in the context of probabilistic classifiers based on the rough set reducts [7] and Naïve Bayes approach. First step towards heterogeneity is by permitting different types of concepts to be used at various levels in hierarchy, but retaining uniformity across a single layer. This creates typical layered learning model [11]. Finally, we may remove all restrictions on the uniformity of models in the neighboring nodes. In this way we produce a more general but harder to control structure.

Synchronous vs. asynchronous. This issue is concerned with the layout of connections between nodes. If it has easily recognizable layered structure we regard it to be synchronized. In other words, we can analyze the hierarchical structure in a level-by-level manner and, consequently, have an ability to clearly indicate the level of abstraction for composite concepts. If we permit the connections to be established on less restrictive basis, the synchronization is lost. Then, the nodes from non-consecutive levels may interact and the whole idea of simple-to-compound precedence of concepts becomes less usable.

The layouts of classifier networks for various levels of homogeneity and synchronization are illustrated in figure 1. In fact, at the moment we are roughly capable of tackling the simplest case of homogeneous and synchronized network corresponding to figure 1a. The partly homogeneous, synchronized architecture that we are attempting to formalize in this paper is shown in figure 1b. Figures 1c and 1d represent the harder cases.

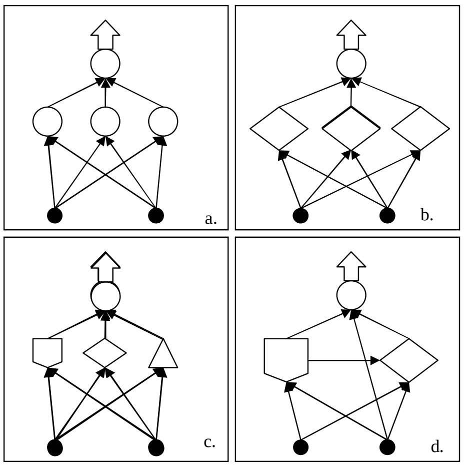

Fig. 1. Examples of network layout: a. both synchronized and homogeneous; b. synchronized and partly heterogeneous; c. synchronized and heterogeneous; d. neither synchronized nor homogeneous.

4 Hierarchical concept schemes

In this section we present a general notation of feedforward networks transmitting the concepts. Since we restrict ourselves to the two easier architecture cases illustrated by figures 1a and 1b, we can consider the following notion:

Definition 1. *By a* hierarchical concept scheme *we mean a tuple* $(\mathcal{C}, \mathcal{MAP})$. $\mathcal{C} = \{C_1, \ldots, C_n, C\}$ *is a collection of the* concept spaces*, where* C *is called the target concept space. The* concept mappings

$$\mathcal{MAP} = \{map_i : C_i \to C_{i+1} : i = 1, \ldots, n, C_{n+1} = C\} \tag{1}$$

are the functions linking consecutive concept spaces.

Any classifier network corresponds to $(\mathcal{C}, \mathcal{MAP})$, i.e. each i-th layer provides us with the elements of C_i. In case of total homogeneity, we have equalities $C_1 = \ldots = C_n = C$ and $map_1 = \ldots = map_n = identity$. For partly homogeneous architecture, some of the mappings can remain identities but we should also expect non-trivial mappings between the concepts of entirely different nature.

Following the structure of feedforward neural network, we calculate the inputs to each next layer as linear combinations of the concepts from the previous one. In general, we cannot expect the traditional definition of a linear combination to be applied. Still, the intuition says that the labels of connections should somehow express the meaning of the concepts in formation of the new ones.

Definition 2. Feedforward concept scheme *is a triple* $(\mathcal{C}, \mathcal{MAP}, \mathcal{LIN})$*, where*

$$\mathcal{LIN} = \left\{lin_i : 2^{C_i \times W_i} \to C_i : i = 1, \ldots, n\right\} \tag{2}$$

defines generalized linear combinations *over the concept spaces* C_i*. For any* $i = 1, \ldots, n$*,* W_i *denotes the space of the* combination parameters*. If* W_i *is a partial or total ordering, then we interpret its elements as* weights *reflecting the relative importance of particular concepts in construction of the resulting concept.*

Let us denote by $m(i) \in \mathbb{N}$ the number of nodes in the i-th network layer. For any $i = 1, \ldots, n$, the nodes from the i-th and $(i+1)$-th layers are connected by the links labelled with parameters $w_{j(i+1)}^{j(i)} \in W_i$, for $j(i) = 1, \ldots, m(i)$ and $j(i+1) = 1, \ldots, m(i+1)$. For any collection of the concepts $c_i^1, \ldots, c_i^{m(i)} \in C_i$ occurring as the outputs of the i-th network's layer in a given situation, the input to the $j(i+1)$-th node in the $(i+1)$-th layer takes the following form:

$$c_{i+1}^{j(i+1)} = map_i\left(lin_i\left(\left\{\left(c_i^{j(i)}, w_{j(i+1)}^{j(i)}\right) : j(i) = 1, \ldots, m(i)\right\}\right)\right) \tag{3}$$

The way of composing functions within the formula (3) requires, obviously, further discussion. In this paper, we restrict ourselves to the case of figure 2a, where map_i and lin_i are stated separately. However, parameters $w_{j(i+1)}^{j(i)}$ could be also used directly in a *generalized concept mapping*

$$genmap_i : 2^{C_i \times W_i} \to C_{i+1} \tag{4}$$

as shown in figure 2b. These two possibilities reflect construction tendencies described in Section 2. Function (4) can be applied to construction of more compound concepts parameterized by the elements of W_i, while the usage of Definitions 1 and 2 results rather in potential syntactical simplification of the new concepts (which can, however, still become more compound semantically).

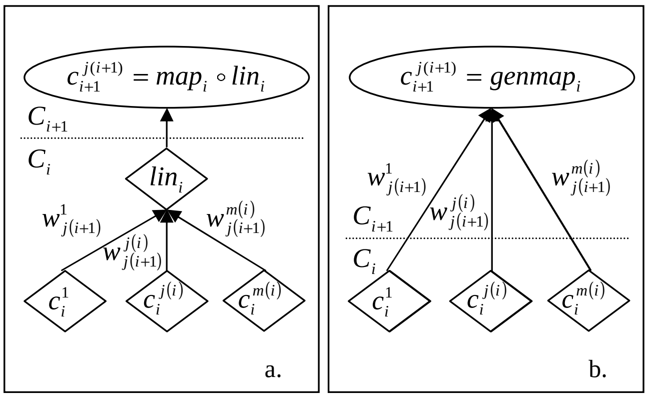

Fig. 2. Production of new concepts in consecutive layers: a. the concepts are first weighted and combined within the original space C_i using function lin_i and then mapped to a new concept in C_{i+1}; b. the concepts are transformed directly to the new space C_{i+1} by using the generalized concept mapping (4).

5 Weighted compound concepts

Beginning with the input layer of the network, we expect it to provide the concepts-signals $c_1^1, \dots, c_1^{m(1)} \in C_1$, which will be then transmitted towards the target layer using (3). If we learn the network related directly to real-valued training sample, then we get $C_1 = \mathbb{R}$, lin_i can be defined as classical linear combination (with $W_i = \mathbb{R}$), and map_i as identity. An example of a more compound concept space originates from our previous studies [9, 10]:

Example 1. Let us assume that the input layer nodes correspond to various classifiers and the task is to combine them within a general system, which synthesizes the input classifications in an optimal way. For any object, each input classifier induces possibly incomplete vector of beliefs in the object's membership to particular decision classes. Let DEC denote the set of decision classes specified for a given classification problem. By the *weighted decision space* $WDEC$ we mean the family of subsets of DEC with elements labelled by their beliefs, i.e.:

$$WDEC = \bigcup_{X \subseteq DEC} \{(k, \mu_k) : k \in X, \mu_k \in \mathbb{R}\} \tag{5}$$

Any *weighted decision* $\widetilde{\mu} = \{(k, \mu_k) : k \in X_{\widetilde{\mu}}, \mu_k \in \mathbb{R}\}$ corresponds to a subset $X_{\widetilde{\mu}} \subseteq DEC$ of decision classes for which the beliefs $\mu_k \in \mathbb{R}$ are known.

Another example corresponds to the specific classifiers – the sets of decision rules obtained using the methodology of rough sets [7, 12]:

Example 2. Let $DESC$ denote the family of logical descriptions, which can be used to define decision rules for a given classification problem. Every *rule* is labelled with its *description* $\alpha_{rule} \in DESC$ and *decision information*, which takes – in the most general framework – the form of $\widetilde{\mu}_{rule} \in WDEC$. For a new object, we measure its degree of satisfaction of the rule's description (usually zero-one), combine it with the number of training objects satisfying α_{rule}, and come out with the number $app_{rule} \in \mathbb{R}$ expressing the level of rule's *applicability* to this object. As a result, by the *decision rule set space* $RULS$ we mean the family of all sets of elements of $DESC$ labelled by weighted decision sets and the degrees of applicability, i.e.:

$$RULS = \bigcup_{X \subseteq DESC} \{(\alpha, \widetilde{\mu}, app) : \alpha \in X, \widetilde{\mu} \in WDEC, app \in \mathbb{R}\} \tag{6}$$

Definition 3. *By a* weighted compound concept space C *we mean a space of collections of* sub-concepts *from some* sub-concept space S *(possibly from several spaces), labeled with the* concept parameters *from a given space* V*, i.e.:*

$$C = \bigcup_{X \subseteq S} \{(s, v_s) : s \in X, v_s \in V\} \tag{7}$$

For a given $c = \{(s, v_s) : s \in X_c, v_s \in V\}$*, where* $X_c \subseteq S$ *is the* range *of* c*, parameters* $v_s \in V$ *reflect relative importance of sub-concepts* $s \in X_c$ *within* c_i*.*

Just like in case of combination parameters W_i in Definition 2, we can assume a partial or total ordering over the concept parameters. A perfect situation would be then to be able to combine these two kinds of parameters while calculating the generalized linear combinations and observe how the sub-concepts from various outputs of the previous layer fight for their importance in the next one. For the sake of simplicity, we restrict ourselves to the case of real numbers:

Definition 4. *Let the* i*-th network layer correspond to the weighted compound concept space* C_i *based on sub-concept space* S_i *and parameters* $V_i = \mathbb{R}$*. Consider the* $j(i+1)$*-th node in the next layer. We define its input as follows:*

$$\begin{aligned} lin_i\left(\left\{\left(c_i^{j(i)}, w_{j(i+1)}^{j(i)}\right) : j(i) = 1, \ldots, m(i)\right\}\right) = \\ = \left\{\left(s, \sum_{j(i): s \in X_{j(i)}} w_{j(i+1)}^{j(i)} v_s^{j(i)}\right) : s \in \bigcup_{j(i)=1}^{m(i)} X_{j(i)}\right\} \end{aligned} \tag{8}$$

where $X_{j(i)} \subseteq S_i$ *is simplified notation for the range of the weighted compound concept* $c_i^{j(i)}$ *and* $v_s^{j(i)} \in \mathbb{R}$ *denotes the importance of sub-concept* $s \in S_i$ *in* $c_i^{j(i)}$*.*

Formula (8) can be applied both to $WDEC$ and $RULS$. In case of $WDEC$, the sub-concept space equals to DEC. The sum $\sum_{j(i): s \in X_{j(i)}} w_{j(i+1)}^{j(i)} v_s^{j(i)}$ gathers the weighted beliefs of the previous layer's nodes in the given decision class $s \in DEC$. In the case of $RULS$ we do the same with the weighted applicability degrees for the elements-rules belonging to the sub-concept space $DESC \times WDEC$.

6 Activation functions

The possible layout combining the concept spaces DEC, $WDEC$, and $RULS$ with the partly homogeneous classifier network is illustrated by figure 3. Given a new object, we initiate the input layer with the degrees of applicability of the rules in particular rule-sets to this object. After processing with these type of concept along (possibly) several layers, we use the concept mapping function

$$map(ruls) = \left\{ \left(k, \textstyle\sum_{(\alpha,\widetilde{\mu},app)\in ruls: k\in X_{\widetilde{\mu}}} app \cdot \mu_k \right) : k \in \bigcup_{(\alpha,\widetilde{\mu},app)\in ruls} X_{\widetilde{\mu}} \right\} \tag{9}$$

that is we simply summarize the beliefs (weighted by the rules' applicability) in particular decision classes. Similarly, we finally map the weighted decision to the decision class, which is assigned with the highest resulting belief.

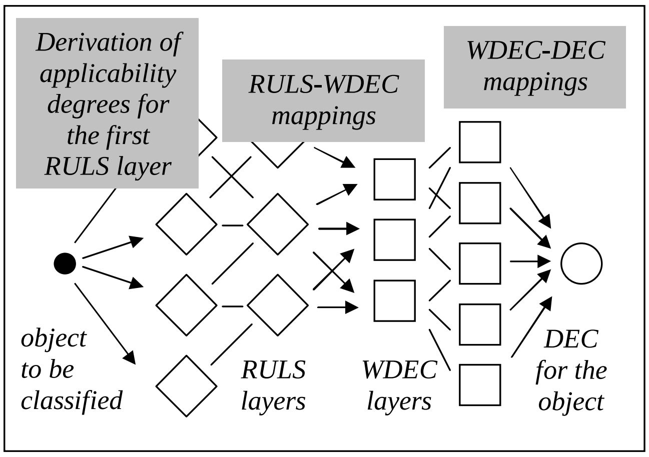

Fig. 3. The network-based object classification: the previously trained decision rule sets are activated by an object by means of their applicability to its classification; then the rule set concepts are processed and mapped to the weighted decisions using function (9); finally the most appropriate decision for the given object is produced.

The intermediate layers in figure 3 are designed to help in voting among the classification results obtained from particular rule sets. Traditional rough set approach (cf. [7]) assumes specification of a fixed voting function, which – in our terminology – would correspond to the direct concept mapping from the first $RULS$ layer into DEC, with no hidden layers and without possibility of tuning the weights of connections. An improved adaptive approach (cf. [12]) enables us to adjust the rule sets, although the voting scheme still remains fixed. In the same time, the proposed method provides us with a framework for tuning the weights and, in this way, learning adaptively the voting formula itself.

Still, the scheme based only on generalized linear combinations and concept mappings is not adjustable enough. The reader may check that composition of functions (8) for elements of $RULS$ and $WDEC$ with (9) results in the collapsed single-layer structure corresponding to the most basic weighted voting among decision rules. This is exactly what happens with classical feedforward neural network models with no non-linear activation functions translating the signals within particular neurons. Therefore, we should consider such functions as well.

Definition 5. Neural concept scheme *is a quadruple* $(\mathcal{C}, \mathcal{MAP}, \mathcal{LIN}, \mathcal{ACT})$, *where the first three entities are provided by Definitions 1, 2, and*

$$\mathcal{ACT} = \{act_i : C_i \to C_i : i = 2, \dots, n+1\} \tag{10}$$

is the set of activation functions, which can be used to relate the inputs to the outputs within each i-th layer of a network.

It is reasonable to assume some properties of $\mathcal{ACT}$, which would work for the proposed generalized scheme analogously to the classical case. Given a compound concept consisting of some interacting parts, we would like, for instance, to guarantee that a relative importance of those parts remains roughly unchanged. Such a requirement, corresponding to monotonicity and continuity of real functions, is well expressible for weighted compound concepts introduced in Definition 3. Given a concept $c_i \in C_i$ represented as the weighted collection of sub-concepts, we claim that its more important (better weighted) sub-concepts should keep more influence on the concept $act_i(c_i) \in C_i$ than the others.

In [9, 10] we introduced sigmoidal activation function working over decision vectors comparable to the structure of $WDEC$ from Example 1. That function, originated from the studies on monotonic decision measures in [8], can be actually generalized onto any space of compound concepts weighted with real values:

Definition 6. *By* α-sigmoidal activation function *for weighted compound concept space C with the real concept parameters, we mean function $act_C^\alpha : C \to C$ parameterized by $\alpha > 0$ which modifies these parameters in the following way:*

$$act_C^\alpha(c) = \left\{ \left(s, \frac{e^{\alpha \cdot v_s}}{\sum_{(t,v_t) \in c} e^{\alpha \cdot v_t}} \right) : (t, v_t) \in c \right\} \tag{11}$$

By composition of lin_i and map_i with functions act_{i+1}^α modifying the concepts within the entire nodes, we obtain a classification model with a satisfiable expressive and adaptive power. If we apply this kind of function to the rule sets, we modify the rules' applicability degrees by their internal comparison. Such performance cannot be obtained using the classical neural networks with the nodes assigned to every single rule. Appropriate tuning of $\alpha > 0$ results in activation/deactivation of the rules with a relative higher/lower applicability. Similar characteristics can be observed within $WDEC$, where the decision beliefs compete with each other in the voting process (cf. [8]).

The presented framework models also other interesting behaviors. For instance, the decision rules which inhibit influence of other rules (so called *exceptions*) can be easily achieved by negative weights and proper activation functions, what would be hard to emulate by plain, negation-free conjunctive decision rules.

7 Learning in classifier networks

A cautious reader have probably already noticed the arising question about the proper choice of connection weights in the network. The weights are ultimately the component that decides about the performance of entire scheme. As we will try to advocate, it is – at least to some extent – possible to learn them in a manner similar to that of backpropagation technique for neural networks.

Bakpropagation, the way we want to use it here, is a method for reducing the global error of a network by performing local changes in weights' values. The key issue is to have a method for dispatching the value of the network's global error functional among the nodes (cf. [3]). This method, when shaped in the form of an algorithm, should provide the direction of the weight update vector, which is then applied according to the learning coefficient. For the standard neural network model (cf. [2]) this algorithm selects the direction of weight update using the gradient of error functional and the current input. Obviously, numerous versions and modifications of gradient-based algorithm exist.

In the more complicated models which we are dealing with, the idea of backpropagation transfers into the demand for a general method of establishing weight updates. This method should comply to the general principles postulated for the rough-neural models (cf. [5, 6, 12]). Namely, the algorithm for the weight updates should provide a certain form of *mutual monotonicity* i.e. small and local changes in weights should not rapidly divert the behavior of the whole scheme and, at the same time, a small overall network error should result in merely cosmetic changes in the weight vectors.

We do not claim to have discovered the general principle for constructing backpropagation-like algorithms for classifier networks. In case of asynchronous network the idea of projecting back (backpropagating) the error to previous layers has no clear meaning. For synchronized models there is more hope. In [9, 10] we have been able to construct generalization of gradient-based method for the homogeneous schemes based on the weighted decision concept space $WDEC$. The step to partly homogeneous schemes is natural for the class of weighted compound concepts, which can be processed using the same type of activation function. For instance, in case of the scheme illustrated by figure 3, the conservative choice of mappings, which turn to be differentiable and regular, permits direct translation from the previous case. Hence, by small adjustment of the algorithm developed previously, we get a *recipé* for learning the weight vectors.

8 Conclusions

We discussed construction of hierarchical concept schemes aiming at layered learning of mappings between the inputs and desired outputs of classifiers. We proposed a generalized structure of feedforward neural- like network approximating the intermediate concepts in a way similar to traditional neurocomputing approaches. We provided the examples of compound concepts corresponding to the decision rule based classifiers and showed some intuition concerning their processing through the network.

Although we have some experience with neural networks transmitting nontrivial concepts [9, 10], this is definitely the very beginning of more general theoretical studies. The most emerging issue are the extension of the proposed framework onto more advanced structures than the introduced weighted compound concepts, without loosing a general interpretation of monotonic activation functions, as well as relaxation of quite limiting mathematical requirements corresponding to the general idea of learning based on the error backpropagation. We are going to challenge these problems by developing our own theoretical and practical foundations, as well as by referring to other approaches, especially those related to rough mereology [6] and rough neurocomputing [5].

Acknowledgements. Partially supported by various grants from Polish Ministry of Scientific Research and Information Technology. The first author supported by the grant awarded from the Faculty of Science at the University of Regina, as well as by the internal research grant of Polish-Japanese Institute of Information Technology.

References

1. Dietterich, T.: Machine learning research: four current directions. *AI Magazine* **18/4** (1997) pp. 97–136.
2. Hecht-Nielsen, R.: Neurocomputing. Addison-Wesley, New York (1990).
3. le Cun, Y.: A theoretical framework for backpropagation. In: Neural Networks – concepts and theory. IEEE Computer Society Press, Los Alamitos (1992).
4. Lenz, M., Bartsch-Spoerl, B., Burkhard, H.-D., Wess, S. (eds.): Case-Based Reasoning Technology – From Foundations to Applications. LNAI 1400, Springer (1998).
5. Peters, J.F., Szczuka, M.: Rough neurocomputing: a survey of basic models of neurocomputation. In: Proc. of RSCTC'2002. LNAI 2475, Springer (2002) pp. 309–315.
6. Polkowski, L., Skowron, A.: Rough Mereology in Information systems. A Case Study: Qualitative Spatial Reasoning. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.): Rough Set Methods and Applications. Physica-Verlag, Heidelberg (2000).
7. Skowron, A., Pawlak, Z., Komorowski, J., Polkowski, L.: A rough set perspective on data and knowledge. In: Kloesgen, W., Żytkow, J. (eds.): Handbook of KDD. Oxford University Press (2002) pp. 134–149.
8. Ślęzak, D.: Normalized decision functions and measures for inconsistent decision tables analysis. *Fundamenta Informaticae* **44/3** (2000) pp. 291–319.
9. Ślęzak, D., Wróblewski, J., Szczuka, M.: Neural Network Architecture for Synthesis of the Probabilistic Rule Based Classifiers. *ENTCS* **82.4**, Elsevier (2003) `http://www.elsevier.nl/locate/entcs/volume82.html`
10. Ślęzak, D., Wróblewski, J., Szczuka, M.: Constructing Extensions of Bayesian Classifiers with use of Normalizing Neural Networks. In: Zhong, N., Raś, Z., Tsumoto, S., Suzuki, E. (eds.), Proc. of ISMIS'2003. LNAI 2871, Springer (2002) pp. 408–416.
11. Stone, P.: Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer. MIT Press, Cambridge MA (2000).
12. Wróblewski, J.: Adaptive aspects of combining approximation spaces. In: Pal, S.K., Polkowski, L., Skowron, A. (eds.): Rough-Neural Computing: Techniques for Computing with Words. Cognitive Technologies Series, Springer, Heidelberg (2003) pp. 139–156.