21

Feedforward Concept Networks*

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@uregina.ca, szczuka@mimuw.edu.pl,
jakubw@pjwstk.edu.pl

Summary. The paper presents an approach to construction of hierarchical structures of data based concepts (granules), extending the idea of feedforward neural networks. The operations of processing the concept information and changing the concept specification through the network layers are discussed. Examples of the concepts and their connections are provided with respect to the case study of learning hierarchical rule based classifiers from data. The proposed methods are referred to the foundations of granular and rough-neural computing.

21.1 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 (which can be also referred to as granules of knowledge [11, 13, 14]). We address these structures as *feedforward concept networks*, which can be regarded as a special case of hierarchical structures developed within the *rough-neural computing* (RNC) methodology [6, 8, 9]. In particular, we consider

*Supported by grant 3T11C00226 from the Polish Ministry of Scientific Research and Information Technology. The first author also supported by the grant from the Faculty of Science, the University of Regina.

classifier networks, where the input concepts correspond to the classified objects' 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 (granules) is not direct but based on the internal layers of intermediate elements, which help in more reliable transition from the basic information to possibly compound classification goal.

We strive against formalization of our approach with use of analogies rooted in other areas, such as *artificial neural networks* [3, 4], *ensembles of classifiers* [2, 12, 21], and *layered learning* [20]. We show how the presented ideas can be exploited within wider frameworks of rough-neural and granular computing. We also make an effort to provide examples of actual models outlined in our earlier, application-oriented papers [18, 19].

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.

21.2 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.

Such an informal definition of a concept space can be referred to the notion of an *information granule system* $S = (G, R, Sem)$, where G is a set of parameterized formulas called *information granules*, R is a (parameterized) relation structure, and Sem is the semantics of G in R (cf. [14]). In our approach, we focus especially on the concept parametrization and ability of parameterized construction of new concepts from the others. In this sense, our understanding of a concept space can be regarded as equivalent to information granule system and the terms *concept* and *granule* can be used exchangeably.

Let a concept represent an 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 the case-based reasoning (cf. [5]), layered learning (cf. [20]), as well as rough mereology [10] and rough neural-computing [6, 8, 9], where we want to approximate target concepts step by step, using the simpler concepts that are easier to learn directly from data.

Construction of simple concepts from the advanced ones.

It can be considered for the synthesis of classifiers, where we start from compound concepts (granules) 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 [9, 11].

The first goal corresponds to generalization of simple concepts while the second – to instantiation of general concept in a simpler, more specialized concept (cf. [16]). 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 scenario to synthesize those compound specifications towards obtaining the final response of the classifier network.

21.3 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 type of concepts (granules) to be used. In the simplest case each node implements the same type of mapping. We have studied such a fully homogeneous system in [18, 19] to express probabilistic classifiers based on the rough set reducts [12] 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 [20]. 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 21.1. The simplest case of homogeneous and synchronized network corresponds to Figure 21.1a. The partly homogeneous, synchronized architecture that we are attempting to formalize in this paper is shown in Figure 21.1b. Figures 21.1c and 21.1d represent the harder cases.

One can see that there are also other cases possible. For instance, we can consider asynchronous but homogeneous network described in [1], where the nodes correspond semantically to the complex concepts we want to approximate although syntactically the operations within the nodes remain of the same type, regardless of whether those nodes represent the advanced of very initial concepts.

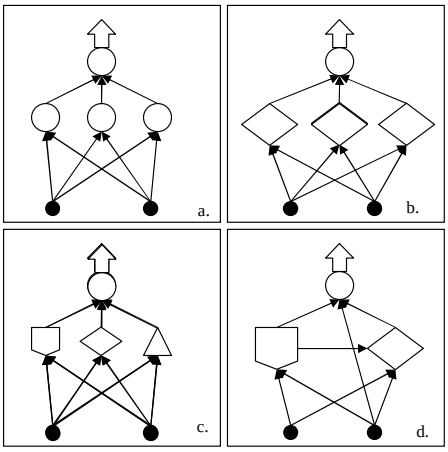

Fig. 21.1. Examples of network layout: a. both synchronized and homogeneous; b. synchronized and partly heterogeneous; c. synchronized and heterogeneous; d. neither synchronized nor homogeneous.

21.4 Hierarchical concept schemes

In this section we present a general notation for feedforward networks transmitting the concepts. Since we restrict ourselves to the two easier architecture cases illustrated by Figures 21.1a and 21.1b, 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 (information granule systems), where* C *is called the target concept space. The* concept mappings

$$\mathcal{MAP} = \{map_i : C_i \to C_{i+1} : i = 1, \ldots, n, C_{n+1} = C\} \tag{21.1}$$

are the functions linking consecutive concept spaces.

We assume that any *feedforward concept 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 $C_1 = \cdots = C_n = C$ and $map_1 = \cdots = map_n = identity$. 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, where $C_i \neq C_{i+1}$.

Following the structure of feedforward neural network, we calculate the inputs to each next layer as 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 level of concepts' importance in formation of the new ones. We refer to this intuition in terms of so called generalized linear combinations:

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

$$\mathcal{LIN} = \left\{ lin_i : 2^{C_i \times W_i} \to C_i : i = 1, \ldots, n \right\} \tag{21.2}$$

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 labeled with parameters $w^{j(i)}_{j(i+1)} \in W_i$, for $j(i) = 1, \ldots, m(i)$ and $j(i+1) = 1, \ldots, m(i+1)$. For any collection of the concepts $c^1_i, \ldots, c^{m(i)}_i \in C_i$ 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:

$$c^{j(i+1)}_{i+1} = map_i \left(lin_i \left(\left\{ \left(c^{j(i)}_i, w^{j(i)}_{j(i+1)} \right) : j(i) = 1, \ldots, m(i) \right\} \right) \right) \tag{21.3}$$

The way of composing functions within the formula (21.3) requires, obviously, further discussion. In this paper, we restrict ourselves to the case of Figure 21.2a, where map_i and lin_i are stated separately. However, parameters $w^{j(i)}_{j(i+1)}$ could be also used directly in a *generalized concept mapping*

$$genmap_i : 2^{C_i \times W_i} \to C_{i+1} \tag{21.4}$$

as shown in Figure 21.2b. These two possibilities reflect construction tendencies described in Section 21.2. Function (21.4) 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).

One can see that function $genmap$ and the corresponding illustration 21.2b refer directly to the ideas of synthesizing concepts (granules, standards, approximations) known from rough-neural computing, rough mereology, and the theory of approximation spaces (cf. [6, 11, 14]). On the other hand, splitting $genmap$'s functionality, as proposed by formula (21.3) and illustrated in 21.2a, provides us with a framework more comparable to the original artificial neural networks and their supervised learning capabilities (cf. [19, 18]).

21.5 Weighted compound concepts

Beginning with the input layer of the network, we expect it to provide the concepts-signals $c_1^1, \ldots, c_1^{m(1)} \in C_1$, which will be then transmitted towards the target layer using (21.3). 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 [18, 19]:

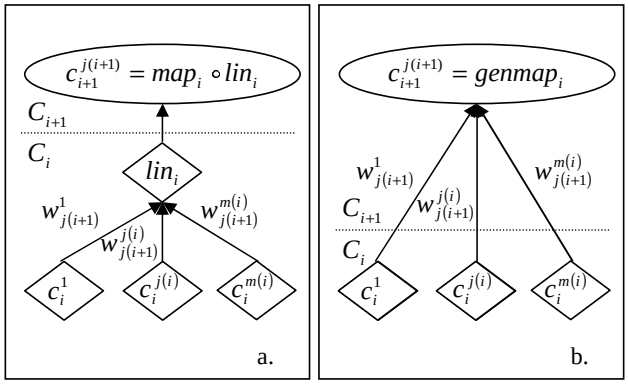

Fig. 21.2. Production of new concepts in consecutive layers: a. the concepts are first weighted and combined within the original space C_i using function lin_i and then mapped to a new concept in C_{i+1}; b. the concepts are transformed directly to the new space C_{i+1} by using the generalized concept mapping (21.4).

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 labeled by their beliefs, i.e.:

$$WDEC = \bigcup_{X \subseteq DEC} \{(k, \mu_k) : k \in X, \mu_k \in \mathbb{R}\} \tag{21.5}$$

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 [12, 21]. The way of parametrization is comparable to the proceedings with classification granules in [11, 14].
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 labeled 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 $DESC$ labeled by weighted decision sets and the degrees of applicability, i.e.:

$$RULS = \bigcup_{X \subseteq DESC} \{(\alpha, \widetilde{\mu}, app) : \alpha \in X, \widetilde{\mu} \in WDEC, app \in \mathbb{R}\} \tag{21.6}$$

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.:*

$$C = \bigcup_{X \subseteq S} \{(s, v_s) : s \in X, v_s \in V\} \tag{21.7}$$

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 further restrict ourselves to the case of real numbers, as stated by Definition 4. However, in general W_i does not need to be $\mathbb{R}$. Let us consider a classifier network, similar to Example 2, where decision rules are described by parameters of accuracy and importance (initially equal to their support). A concept transmitted by network refers to rules matched by an input object. The generalized linear combination of such concepts may be parameterized by vectors $(w, \theta) \in W_i$ and defined as a union of rules, where importance is expressed by w and θ states a threshold for the rules' accuracy.

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:*

$$\begin{aligned} lin_i\left(\left\{\left(c_i^{j(i)}, w_{j(i+1)}^{j(i)}\right) : j(i) = 1, \ldots, m(i)\right\}\right) = \\ = \left\{\left(s, \sum_{j(i):s \in X_{j(i)}} w_{j(i+1)}^{j(i)} v_s^{j(i)}\right) : s \in \bigcup_{j(i)=1}^{m(i)} X_{j(i)}\right\} \end{aligned} \tag{21.8}$$

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 (21.8) 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 $DESC \times WDEC$.

It is interesting to compare our method of the parameterized concept transformation with the way of proceeding with classification granules and decision rules in the other rough set based approaches [11, 12, 14, 21]. Actually, at this level, we do not provide anything novel but rewriting well known examples within a more unified framework. A more visible difference can be observed in the next section, where we complete our methodology.

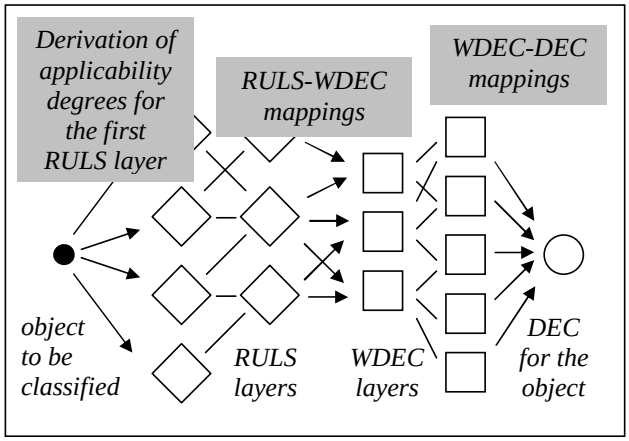

Fig. 21.3. The network-based object classification: the previously trained decision rule sets are activated by an object by means of their applicability to its classification; then the rule set concepts are processed and mapped to the weighted decisions using function (21.9); finally the most appropriate decision for the given object is produced.

21.6 Activation functions

The possible layout combining the concept spaces DEC, $WDEC$, and $RULS$ with the partly homogeneous classifier network is illustrated by Figure 21.3. 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 this type of concept along (possibly) several layers, we use the concept mapping function

$$map(ruls) = \left\{ \left(k, \sum_{(\alpha,\widetilde{\mu},app) \in ruls : k \in X_{\widetilde{\mu}}} app \cdot \mu_k \right) : k \in \bigcup_{(\alpha,\widetilde{\mu},app) \in ruls} X_{\widetilde{\mu}} \right\} \tag{21.9}$$

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.

The intermediate layers in Figure 21.3 are designed to help in voting among the classification results obtained from particular rule sets. Traditional rough set approach (cf. [12]) 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. [21]) 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 (cf. [6, 11, 14]).

Still, the scheme based only on generalized linear combinations and concept mappings is not adjustable enough. The reader may check that composition of functions (21.8) for elements of $RULS$ and $WDEC$ with (21.9) 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*

$$\mathcal{ACT} = \{act_i : C_i \to C_i : i = 2, \ldots, n+1\} \tag{21.10}$$

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 [18, 19] we introduced sigmoidal activation function working on probability vectors comparable to the structure of $WDEC$ in Example 1. That function, originated from the studies on monotonic decision measures in [15], 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:*

$$act_C^\alpha(c) = \left\{ \left(s, \frac{e^{\alpha \cdot v_s}}{\sum_{(t,v_t)\in c} e^{\alpha \cdot v_t}} \right) : (t, v_t) \in c \right\} \tag{21.11}$$

By composition of lin_i and map_i, which specify the concepts $c_i^{j(i+1)} \in C_{i+1}$ as inputs to the nodes in the $(i+1)$-th layer, 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. [15]).

The presented framework allows for modeling 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. Further research is needed to compare the capabilities of the proposed construction with other hierarchical approaches [6, 10, 9, 20].

21.7 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 the case of standard neural networks.

Backpropagation, 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. [4]). 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. [3]) 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. [8, 21]). 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. The need of introducing automatic backpropagation-like algorithms to rough-neural computing were addressed recently in [6]. It can be referred to some already specified solutions like, e.g., the one proposed for rough-fuzzy neural networks in [7]. Still, general framework for RNC is missing, where a special attention must be paid on the issue of interpreting and calculating partial error derivatives with respect to the complex structures' parameters.

We do not claim to have discovered the general principle for constructing backpropagation-like algorithms for the concept (granule) networks. Still, in [18, 19] we have been able to construct generalization of gradient-based method for the homogeneous neural concept schemes based on the 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 21.3, 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.

An example of two-dimensional weights $(w, \theta) \in W_i$ proposed in Section 21.4 is much harder to translate into backpropagation language. One of the most important features of classical backpropagation algorithm is that we can achieve the local minimum of an error function (on a set of examples) by local, easy to compute, change of the weight value. It does not remain easy for two real-valued parameters instead of one. Moreover, parameter θ is a rule threshold (fuzzified by a kind of sigmoidal characterisitcs to achieve differentiable model) and, therefore, by adjusting its value we are switching on and off (almost, up to the proposed sigmoidal function) entire rules, causing dramatic error changes. This is an illustration of the problems arising when we are dealing with more complicated parameter spaces – In many cases we have to use dedicated, time-consuming local optimization algorithms.

Yet another issue is concerned with the second ,,tooth" of backpropagation: transmitting the error value backward the network. The question is how to modify the error value due to connection weight, assuming that the weight is generalized (e.g. the vector as above). The error value should be translated into value compatible with the previous layer of classifiers, and should be useful for an algorithm of parameters modification. It means that information about error transmitted to the previous layer can be not only a real-valued signal, but e.g. a complete description of each rule's positive or negative contribution to the classifier performance in the next layer.

21.8 Conclusions

We have 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 non-trivial concepts [18, 19], this is definitely the very beginning of more general theoretical studies. The most emerging issue is the extension of 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 theoretical and practical foundations, as well as by referring to other approaches, especially those related to rough-neural computing [6, 8, 9].

References

1. Bazan, J., Nguyen, S.H., Nguyen, H.S., Skowron, A.: Rough Set Methods in Approximation of Hierarchical Concepts. In: Proc. of RSCTC'2004. LNAI**3066**, Springer Verlag (2004) pp. 346–355
2. Dietterich, T.: Machine learning research: four current directions. AI Magazine **18/4** (1997) pp. 97–136.
3. Hecht-Nielsen, R.: Neurocomputing. Addison-Wesley (1990).
4. le Cun, Y.: A theoretical framework for backpropagation. In: Neural Networks – concepts and theory. IEEE Computer Society Press (1992).
5. Lenz, M., Bartsch-Spoerl, B., Burkhard, H.-D., Wess, S. (eds.): Case-Based Reasoning Technology: From Foundations to Applications. LNAI **1400**, Springer (1998).
6. Pal, S.K., Peters, J.F., Polkowski, L., Skowron, A.: Rough-Neural Computing: An Introduction. In: S.K. Pal, L. Polkowski, A. Skowron (eds.), Rough-Neural Computing. Cognitive Technologies Series, Springer (2004) pp. 15–41.
7. Pedrycz, W., Peters, J.F.: Learning in fuzzy Petri nets. In: J. Cardoso, H. Scarpelli (eds.), Fuzziness in Petri Nets. Physica (1998) pp. 858–886.
8. 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.
9. Polkowski, L., Skowron, A.: Rough-neuro computing. In: W. Ziarko, Y.Y. Yao (eds.), Proc. of RSCTC'2000. LNAI **2005**, Springer (2001) pp. 57–64.
10. Polkowski, L., Skowron, A.: Rough mereological calculi of granules: A rough set approach to computation. Computational Intelligence, **17/3** (2001) pp. 472–492.
11. Skowron, A.: Approximate Reasoning by Agents in Distributed Environments. Invited speech at IAT'2001. Maebashi, Japan (2001).
12. Skowron, A., Pawlak, Z., Komorowski, J., Polkowski, L.: A rough set perspective on data and knowledge. In: W. Kloesgen, J. Żytkow (eds.), Handbook of KDD. Oxford University Press (2002) pp. 134–149.
13. Skowron, A., Stepaniuk, J.: Information granules: Towards foundations of granular computing. International Journal of Intelligent Systems **16/1** (2001) pp. 57–86.
14. Skowron, A., Stepaniuk, J.: Information Granules and Rough-Neural Computing. In: S.K. Pal, L. Polkowski, A. Skowron (eds.), Rough-Neural Computing. Cognitive Technologies Series, Springer (2004) pp. 43–84.
15. Ślęzak, D.: Normalized decision functions and measures for inconsistent decision tables analysis. Fundamenta Informaticae **44/3** (2000) pp. 291–319.
16. Ślęzak, D., Szczuka, M., Wróblewski, J.: Harnessing classifier networks – towards hierarchical concept construction. In: Proc. of RSCTC'2004, Springer (2004).
17. Ślęzak, D., Wróblewski, J.: Application of Normalized Decision Measures to the New Case Classification. In: W. Ziarko, Y. Yao (eds.), Proc. of RSCTC'2000. LNAI **2005**, Springer (2001) pp. 553–560.
18. Ślęzak, D., Wróblewski, J., Szczuka, M.: Neural Network Architecture for Synthesis of the Probabilistic Rule Based Classifiers. ENTCS **82/4**, Elsevier (2003).
19. Ślęzak, D., Wróblewski, J., Szczuka, M.: Constructing Extensions of Bayesian Classifiers with use of Normalizing Neural Networks. In: N. Zhong, Z. Raś, S. Tsumoto, E. Suzuki (eds.), Proc. of ISMIS'2003. LNAI **2871**, Springer (2002) pp. 408–416.
20. Stone, P.: Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer. MIT Press, Cambridge MA (2000).
21. Wróblewski, J.: Adaptive aspects of combining approximation spaces. In: S.K. Pal, L. Polkowski, A. Skowron (eds.), Rough-Neural Computing. Cognitive Technologies Series, Springer (2004) pp. 139–156.