# Harnessing Classifier Networks – Towards Hierarchical Concept Construction

Dominik Ślęzak[1,2], Marcin S. 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

In the standard approach to classification, we use hypothesis formation (learning) algorithm to find a possibly direct mapping from the input values to decisions. Such an approach does not always result in success. We address the situation when the desired solution should have an internal structure. Although possibly hard to find and learn, such architecture provide 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 concept reflects the final classification. The relationship between 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, 10], and layered learning [9]. We present the formalism that describes our classifier networks and illustrate them with examples of actual models.

## 2 Hierarchical Learning and Classification

In general, a **concept** is an element of a parameterized concept space. By a proper setting of these parameters we choose the right concept. In our approach, a concept represents a basic element acting on the information originating from other concepts or the data source. As an analogy, consider the case of neural networks (NN), where a concept corresponds to a signal transmitted through a neuron. Dependencies between concepts and their importance are represented by weighted node connections. Similarly to the feedforward NN, 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 [9], as well as rough mereology [6] and rough neurocomputing [5], where we approximate target concepts step by step, using the simpler ones that are easier to learn 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 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.

The above are not the only possible types of constructions. In a classification problem, decision classes can have a compound semantics requiring gradual specification corresponding to the first type of construction. When considering hierarchical structures at the general level one has to make choices with respect to homogeneity and synchronization:

**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 [8]. First step towards heterogeneity is by permitting different types of concepts to be used at various levels in hierarchy, but retaining any single layer uniform (the layered learning model [9]). Finally, we may remove all restrictions on the uniformity of the neighboring nodes; such structure is more general but harder to control.

**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. 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. Restricting ourselves to the two easier architectures, we can consider the following notion concerning feedforward networks transmitting the concepts:

**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 $\mathcal{MAP} = \{map_i : C_i \to C_{i+1} : i = 1, \ldots, n, C_{n+1} = C\}$ *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 $C_i = C$ and $map_i = identity$. For partly homogeneous case the mappings may be nontrivial.

Following the structure of feedforward neural network, the inputs to each next layer are linear combinations of the concepts from the previous one. In general, we cannot apply the traditional definition of a linear combination.

**Definition 2.** Feedforward concept scheme *is a triple* $(\mathcal{C}, \mathcal{MAP}, \mathcal{LIN})$*, where* $\mathcal{LIN} = \{lin_i : 2^{C_i \times W_i} \to C_i : i = 1, \ldots, n\}$ *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 $w_{j(i+1)}^{j(i)} \in W_i$, 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:

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

Consider the case of Fig. 1a, where $map_i$ and $lin_i$ are stated separately. Parameters $w_{j(i+1)}^{j(i)}$ could be also used directly in a *generalized concept mapping* $genmap_i : 2^{C_i \times W_i} \to C_{i+1}$ as in figure 1b. These two possibilities reflect construction tendencies described in Section 2. Function $genmap_i$ 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.

## 3 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 (1). 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.

*Example 1.* [8] 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 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. 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.

*Example 2.* Let $DESC$ denote the family of logical descriptions, which can be used to define rough-set-based decision rules for a given classification problem [10]. 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 $\alpha_{rule}$, 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.

**Definition 3.** *By a* weighted compound concept space $C$ *we mean a space of collections of* sub-concepts *from some* sub-concept space $S$*, labeled with the* concept parameters *from a given space* $V$*, i.e.,* $C = \bigcup_{X \subseteq S} \{(s, v_s) : s \in X, v_s \in V\}$. *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* $c_i$*.*

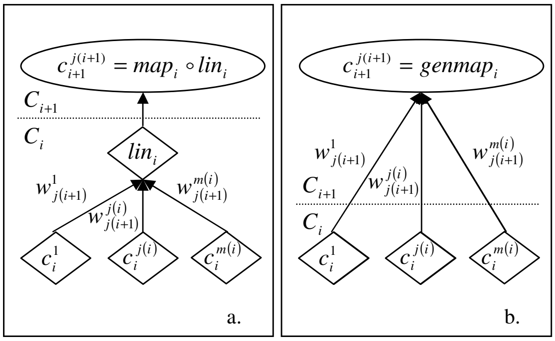

**Fig. 1.** Production of new concepts in consecutive layers.

Just like in case of combination parameters $W_i$ in Definition 2, we can assume a partial or total ordering over the concept parameters.

**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:*

$$\begin{aligned} lin_i\left(\left\{\left(c_i^{j(i)}, w_{j(i+1)}^{j(i)}\right) : j(i) = 1, \ldots, m(i)\right\}\right) = \\ = \left\{\left(s, \sum_{j(i): s \in X_{j(i)}} w_{j(i+1)}^{j(i)} v_s^{j(i)}\right) : s \in \bigcup_{j(i)=1}^{m(i)} X_{j(i)}\right\} \end{aligned} \tag{2}$$

*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 (2) 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 $DESC \times WDEC$.

## 4 Activation Functions

In rule-based classifier network, we initiate the input layer with the degrees of applicability of the rules in particular rule-sets to a new object. After processing with these type of concept along (possibly) several layers, we use the concept mapping function

$$map(ruls) = \left\{ \left( k, \sum_{(\alpha,\widetilde{\mu},app) \in ruls : k \in X_{\widetilde{\mu}}} app \cdot \mu_k \right) : k \in \bigcup_{(\alpha,\widetilde{\mu},app) \in ruls} X_{\widetilde{\mu}} \right\} \quad (3)$$

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 are designed to help in voting among the classification results obtained from particular rule sets. Traditional rough set approach 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 [10] enables us to adjust the rule sets, but the voting scheme is still fixed. The new method provides us with a framework for tuning the weights and learning adaptively the voting formula.

Still, the scheme based only on generalized linear combinations and concept mappings is not adjustable enough. The reader may check that composition of functions (2) for elements of $RULS$ and $WDEC$ with (3) 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 linear activation functions.

**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* $\mathcal{ACT} = \{act_i : C_i \to C_i : i = 2, \ldots, n+1\}$ *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 a kind of monotonicity of $\mathcal{ACT}$ (a relative importance of parts of concept remains roughly unchanged). It is 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 [8] we introduced sigmoidal activation function, which can be generalized as $act_C^{\alpha}(c) = \left\{ \left( s, e^{\alpha \cdot v_s} / \sum_{(t,v_t) \in c} e^{\alpha \cdot v_t} \right) : (t, v_t) \in c \right\}$

(see [7]). By composition of $lin_i$ and $map_i$ with functions $act^{\alpha}_{i+1}$ 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. Moreover, the decision rules which inhibit influence of other rules (so called *exceptions*) can be easily achieved by negative weights and proper activation functions.

## 5 Learning in Classifier Networks

The proper choice of connection weights in the network can be learned similarly to backpropagation technique for neural networks. Backpropagation 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 [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 (see [2] for classical setting). 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 [5, 6, 10]. 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 [8] 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.

## 6 Conclusions

Although we have some experience with neural networks transmitting non-trivial concepts [8], this is definitely the very beginning of more general theoretical studies. The most emerging issue is the extension of the proposed framework onto more advanced structures than the introduced weighted compound concepts, without losing a general interpretation of monotonic activation functions, as well as relaxation of some quite limiting mathematical requirements. We are going to challenge these problems in further works.

## Acknowledgements

Supported by grant 3T11C00226 from Polish Ministry of Scientific Research. Authors also supported by the Faculty of Science at the University of Regina, and 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., Szczuka, M.: Rough neurocomputing: a survey of basic models of neurocomputation. In: Proc. of RSCTC'02. 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. Ślęzak, D.: Normalized decision functions and measures for inconsistent decision tables analysis. *Fundamenta Informaticae* **44/3** (2000) pp. 291–319.
8. 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.
9. Stone, P.: Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer. MIT Press, Cambridge MA (2000).
10. 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. Springer, Heidelberg (2003) pp. 139–156.