Neural Expert Systems

Jiří Šíma
Department of Theoretical Informatics
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2,
182 07 Prague 8,
Czech Republic
e-mail: sima@uivt.cas.cz

Acknowledgement

I wish to thank Dr. O. Kufudaki and Dr. J. Hořejš for their willingness to spend time with me discussing various problems. Further, I would like to thank Dr. A. Grosmanová, who prepared the clinical data for the neural expert system application. And I am also very thankful to my colleague and friend Dr. R. Neruda, who created the bigger part of the program EXPSYS, concerning the interaction between the user and the system.

Neural Expert Systems

Abstract

The advantages and disadvantages of classical rule-based and neural approaches to expert system design are complementary. We propose a strictly neural expert system architecture, that enables the creation of the knowledge base automatically, by learning from example inferences. For this purpose we employ a multi-layered neural network, trained with generalized back propagation for interval training patterns, that also makes the learning of patterns with irrelevant inputs and outputs possible. We eliminate the disadvantages of the neural approach by enriching the system with the heuristics to work with incomplete information, and to explain the conclusions. The structure of the expert attributes is optional, and a user of the system can define the types of inputs and outputs (real, integer, scalar type, and set), and the manner of their coding (floating point, binary, and unary codes). We have tested our neural expert system on several non-trivial real-world problems (e.g., the diagnostics and progress prediction of hereditary muscular disease), and the results are very good.

Key words: expert system, knowledge representation, multi-layered neural network, back propagation, interval neuron function, incomplete information, explanation

1 Introduction

1.1 Rule-Based and Neural Expert Systems

The great hopes originally placed upon expert systems are still far from being fulfilled. Research has brought only limited applicable results. The quest for a tool, which could make the representation of large amount of knowledge possible, as well as consistent and effectively usable, is one of the basic problems of artificial intelligence. Basically, there are two main approaches to expert system design.

Knowledge representation in conventional expert systems is based on rules. This means that a human expert is needed to extract regularities from his experience, and to express them in the comprehensible, explicit, form of rules. Due to the explicitness of the knowledge, the system has perfect explanation abilities, and works well with inaccurate and incomplete information — for example when using some confidence factors. On the other hand, building such a consistent knowledge base is a difficult matter. Editing can produce some side effects. Therefore the average development time is 12 to 18 months, and large systems require careful design.

The second main approach is connected with the rapid development of neural network theory. In so-called neural expert systems, the knowledge base is a neural network which is created automatically by a learning algorithm, from a set of example inferences. That is why building this system takes just a few weeks or months, and additional learning is possible. But, large systems need long-term, computationally exacting learning, with uncertain results (Blum and Rivest, 1992; Judd, 1990; Svaytser, 1989; Šíma 1994b, 1994c). The representation of knowledge is based on numerical weights corresponding to the connections among neurons. Meanwhile, there is no general way to identify a purpose to single neurons in neural networks because of the implicit knowledge representation (Hořejš, 1992). Therefore, work with incomplete information, and providing justification of inference is limited or even impossible.

As we have seen above, the advantages and disadvantages of rule-based and neural approaches to the expert system design are complementary. There is a natural tendency to integrate advantages of both. Some general possible approaches to such hybrid systems have been described by Caudill (1991).

1.2 Current State

In current neural network literature, the tension between rule-based and neural approaches to the expert system design is clearly marked. On one hand, there is an explicit knowledge representation which makes the explanation heuristics and work with incomplete information possible, and on the other hand, there are strong learning algorithms which create the knowledge base automatically. The hybrid models proposed in literature are always compromises, sometimes preferring explanation abilities and work with incomplete information to the detriment of an automatic adaptability, and other times preferring a neural network architecture which sacrifices explicit work with information.

One problem in integrating both approaches, is the creation of the knowledge base when only some rules and the example data are available. McMillan, Mozer, and Smolensky (1991); Samad (1988); Yang and Bhargava (1990) proposed some heuristics how to construct the neural knowledge base from explicit rules. Frasconi, Gori, Maggini, and Soda (1991); Handelman, Lane, and Gelfand (1989); Hruska, Kuncicky, and Lacher (1991) dealt with so-called hybrid learning, when neural network weights are initialized from explicit rules (e.g., using linear programming), and learning from examples is then viewed as a refinement process to cover uncertainty. The opposite approach is considered by Kasabov (1991), who incorporates rules into an already trained neural network.

Another possible approach is to create additional rules from example data, with the aid of neural learning, in order to complete the knowledge base of the rule-based system. Saito and Nakano (1988) proposed the heuristics of rule extraction from data and from a trained neural network. Enbutsu, Baba, and Hara (1991) introduced a so-called causal index as a measure of the causal relationship between input and output neurons. They used this index to extract rules from a trained neural network. A similar idea with a relation factor was introduced by Krysiak (1991).

Further problems appear when we encode incomplete and irrelevant information in neural networks (Drucker, 1990). We proposed a solution for this issue, which is based on interval neuron function (Šíma, 1992b). It is also possible to teach the neural network to work with incomplete information by including incomplete example data into the training set instead of using special inference heuristics (Šíma, 1992a). Gallant (1987, 1988a) precisely examined a similar idea concerning inaccurate information. He generated training data with respect to conclusion frequency, importance, and input data noise.

The classical paper concerning neural expert systems (Gallant, 1988b, 1993), considers a simple neural net model with better interpretation of knowledge representation, but with a weaker learning algorithm. The heuristics of working with incomplete information, and explanation, are proposed directly for this neural net model. Our proposal is based on a similar approach.

Recently, first published results concerning successful neural expert system applications (e.g. Baxt, 1990; Mozer, 1987) have appeared.

1.3 Neural Expert System Proposal

In this paper, we propose a neural expert system architecture inspired by Gallant (1988). We do not compromise the strictly neural approach that enables the creation of the knowledge base automatically by learning from example inferences. Therefore, the multi-layered neural network is employed. We generalized the back propagation learning algorithm for interval training patterns, that also makes the learning of patterns with irrelevant inputs and outputs possible (section 3). The disadvantages of the neural approach are eliminated by enriching the system with the heuristics for working with incomplete information (section 4), and for explaining the conclusions (section 5). The structure of the expert attributes is optional, and a user of the system can define the types of inputs and outputs (real, integer, scalar type, and set) and the manner of their coding (floating point, binary and unary codes) (section 2). We have tested our expert system on several non-trivial, real-world problems (e.g., the diagnostics and progress prediction of hereditary muscular disease), and the results are very good (section 6).

2 Expert Inputs and Outputs

In the particular design of the diagnostic expert system, it is necessary to choose the expert inputs and outputs which should best render a problem solved by this system. Our proposal (Šíma and Neruda, 1992a, 1992b) of the neural expert system makes various types of expert inputs and outputs possible as well as various ways of their coding with respect to the needs of a particular application. Now we will describe the definitions of the types, and their representations in detail.

2.1 Types of Expert Inputs and Outputs

Let us consider the neural expert system with a number of inputs (e.g. symptoms) and outputs (e.g. diagnoses). These can be of various Pascal-like types: real (integer) number, scalar type with user-defined values, and set. The domain of the real (integer) number is given by a real (integer) interval. The domain of the scalar type is defined by the vector of possible values, and the domain of the set type is determined by the universe of possible elements, given also by the vector of these elements. The example of a simple medical diagnostic expert system could have the following parameters:

```
INPUTS: SORE-THROAT: scalar of (NO,YES)
        TEMPERATURE: real of <36,42>
OUTPUTS: ILLNESS: scalar of (HEALTHY,COLD,ANGINA,INFLUENZA)
         INABILITY-TO-WORK: scalar of (NO,YES)
         MEDICATIONS: set of (DROPS,ASPIRIN,PENICILLIN)
```

2.2 Coding Expert Inputs and Outputs

The values of inputs and outputs of the neural expert system are encoded by analog values of the input and output neuron states in three possible ways — using floating-point, binary, and unary codes. Almost any combination of type and coding is allowed, with the exception of a unary coded real type, which does not make good sense. A binary coded real type is, in fact, an integer type. The floating-point coding requires only one neuron. In the unary and binary codes there are generally more coding neurons with two states (-1 and 1) used, according to the type definition.

In the process of coding, the value of an arbitrary type is first transformed to a numerical value, which we call an index. The description of the index evaluation for individual types follows. The index is directly equal to the value of the real type when floating-point coding is used. To determine the index when coding an integer binary, the lower bound of the interval domain is subtracted from this integer. For the scalar type, the index is determined as the order of the coded value in the domain definition, counting from zero. The index of a coded set corresponds to the value of a binary number. This number is obtained when looking at the universe given by the vector of possible elements in the type definition, like it is a binary integer where the presence of an element in coded set means the binary digit 1 and its absence means 0. The index evaluation is illustrated in Tab.1 for the values of different types taken from our example.

Type	Example Value	Relevant Index $\in \langle p, q \rangle$	Floating Point	Binary Code	Unary Code
real	38^0C	$38 \in \langle 36, 42 \rangle$	$-\frac{1}{3}$	—	—
integer	38^0C	2	—	-1,1,-1	—
scalar	`INFLUENZA`	$3 \in \langle 0, 3 \rangle$	1	1,1	-1,-1,-1,1
set	{ `DROPS`, `PENICILLIN` }	$5 \in \langle 0, 7 \rangle$	$\frac{3}{7}$	1,-1,1	-1,-1,-1,-1, -1,1,-1,-1

Table 1: Example of Coding

The numerical index is then encoded by neuron states using the codes as follows. In the case of floating-point, the index is transformed to the analog value of the relevant neuron state through normalization. Let $\langle p, q \rangle$ be the interval of all possible indexes according to the type definition. We carry out the linear transformation of this interval to the interval $\langle -1, 1 \rangle$. This corresponds to the linear transformation of the index I to the neuron state $\frac{2(I-p)}{q-p} - 1 \in \langle -1, 1 \rangle$. For binary code we need $\lceil \log_2 n \rceil$ neurons where n is the number of possible indexes given by the type definition. The index is then binary encoded and the binary digits 0,1 are represented by neuron states -1,1. Similarly, n neurons are needed for unary coding. Every neuron corresponds to one possible index value. The relevant neuron state is 1 and all remaining neuron states are -1. Examples of coding are also shown in the Tab.1.

From the interpretation above, the natural combinations of the type and coding follow: floating-point coding of the real type, binary coding of the set type, and unary coding of the scalar type.

3 The Neural Knowledge Base

3.1 Example Inferences

An expert formulates his knowledge in the form of typical example inferences. An example inference is a pair composed of the vector of the typical inputs and the corresponding vector of the outputs observed from the expert responses to these inputs. The set of example inferences should cover the basic required regularities of the studied area. Their combining and generalization is left to the learning process.

For example, in medicine the medical documentation about patients can be taken as a basis for the creation of the set of typical example inferences. In this case, the inputs correspond to the outcomes of various examinations, and the outputs are diagnoses or medications recommended by a doctor. Continuing in our example from section 2, the set of example inferences can take the following form:

$$\{\ [\ (\mathtt{NO,39.6}),\ (\mathtt{INFLUENZA,YES,}\{\mathtt{DROPS,ASPIRIN}\})\ ],$$
$$[\ (\mathtt{YES,37.2}),\ (\mathtt{ANGINA,YES,}\{\mathtt{PENICILLIN}\})\ ],\ldots,$$
$$[\ (\mathtt{NO,}\mathit{unknown}),\ (\mathtt{HEALTHY,NO,}\{\mathtt{ASPIRIN}\})\ ]\ \}$$

The knowledge base of our neural expert system is the multi-layered neural network (Rumelhart, Hinton, and Williams, 1986). The input and output layer neurons of this network encode the value of expert inputs and outputs. The manner of coding has been described in section 2.2.

But the problem of how to encode the irrelevant and unknown value of expert inputs and outputs appears. In our example, the unknown input value occurs in the last element of the above-mentioned set of example inferences. To understand the importance of this problem better, let us consider the example of a medical expert system with 50 inputs representing outcomes of all possible medical examinations.

In the particular case, most of them are irrelevant for the determination of the output diagnosis. For example, an X-ray is not needed for stating the diagnosis of angina.

To solve this coding problem, we could create more example inferences substituting all possible values for irrelevant inputs and outputs. But this process leads mostly to an unmanageable amount of data. Therefore, instead of this, we generalize the classical single neuron state to the interval one. Then the irrelevant or even unknown value of an expert input or output is encoded using the full interval neuron state $\langle -1, 1 \rangle$, and the known value is encoded by a one-point interval determined by the coding procedure described in section 2.2.

This way, we encode the set of example inferences to the so-called training set with interval training patterns, which will be used later in the learning process. The interval training pattern is composed of the vector of the interval input neuron states and the corresponding vector of the interval output neuron states. In our example, using natural coding of types, the corresponding interval training pattern to the last above-mentioned example inference follows:

```
[ ( <1,1>,<-1,-1>                           NO
    <-1,1> ),                               unknown
  ( <1,1>,<-1,-1>,<-1,-1>,<-1,-1>,          HEALTHY
    <1,1>,<-1,-1>,                          NO
    <-1,-1>,<1,1>,<-1,-1> ) ]               { ASPIRIN }
```

3.2 Interval Neuron Function

Due to the interval neuron state, we have to generalize the classical neuron function of the multi-layered network to the interval one. The motivation and detailed derivation of the interval neuron function has been described by Šíma (1992b). Here we present just the resulting formulas.

Assume that i is a non-input neuron, whose inputs are connected with neurons from the set $\overline{i}$, and that these connections are evaluated by synaptic weights $w_{ij}, j \in \overline{i}$, and that w_{i0} is a bias. We introduce the predicate FP with the neuron as a parameter, which is true iff this neuron is an output neuron, which encodes an expert output using floating-point. In the new approach, we consider the interval inputs $\langle a_j, b_j \rangle, j \in \overline{i}$ of the neuron i. Then we determine the following interval inner potential $\langle x_i, y_i \rangle$ and the interval state (output) $\langle a_i, b_i \rangle$ for this neuron:

$$
\begin{aligned}
a_i &= \begin{cases} x_i & FP(i) \\ S_i(x_i) & \text{otherwise} \end{cases} \qquad b_i = \begin{cases} y_i & FP(i) \\ S_i(y_i) & \text{otherwise} \end{cases} \\
S_i(x) &= \frac{1 - e^{-\lambda_i x}}{1 + e^{-\lambda_i x}} \quad \neg FP(i) \\
x_i &= w_{i0} + \sum_{j \in \overline{i}} w_{ij} \left(s_i(w_{ij}) a_j + \overline{s}_i(w_{ij}) b_j \right) \\
y_i &= w_{i0} + \sum_{j \in \overline{i}} w_{ij} \left(\overline{s}_i(w_{ij}) a_j + s_i(w_{ij}) b_j \right) \\
s_i(x) &= \frac{1}{1 + e^{-\lambda_i x}} \qquad \overline{s}_i(x) = s_i(-x)
\end{aligned}
\tag{1}
$$

where λ_i is the gain parameter of the transfer function S_i and of the auxiliary signum functions $s_i, \overline{s}_i$ (if $FP(i)$ then set λ_i to 1).

The interval neuron function approximately preserves the idea of the monotony with respect to the interval inclusion. This means, for $\lambda \to \infty$, if any single neuron inputs $z_j \in \langle a_j, b_j \rangle, j \in \overline{i}$ are taken from the interval neuron inputs, then the single neuron output $z_i = S_i \left(w_{i0} + \sum_{j \in \overline{i}} w_{ij} z_j \right)$, computed by classical single neuron function, must fall into the output neuron interval $\langle a_i, b_i \rangle$ computed by the interval neuron function (1), that is $z_i \in \langle a_i, b_i \rangle$.

The generalized interval vector function of the multi-layered neural network is computed gradually, from the input layer via the hidden layers up to the output layer, using the formulas (1) for the interval neuron function.

3.3 Learning Algorithm

For particular neural expert system design the number of hidden layers and the number of neurons in them are estimated with respect to the size of the practical problem. There is still the problem of how to determine the synaptic weights and gain parameters of the interval network function, so that this vector function will be the generalization of the set of interval training patterns acquired by coding of the set of example inferences. For this purpose, we generalize the back propagation learning algorithm for interval training patterns (Šíma, 1992b).

We consider the training set $\{(I_k, O_k)/\, k = 1, \ldots, N\}$, where for the kth training pattern, I_k is the vector of the network input intervals, and O_k is the corresponding required vector of the network output intervals $\langle A_{kv}, B_{kv}\rangle$, where v denotes the relevant output neuron. Let $\langle a_v(I_k), b_v(I_k)\rangle$ be the actual interval state of the output neuron v for network input I_k. We define the partial error function E_k of the kth training pattern as a sum of discrepancies between the required and actual network output intervals. Because of the idea of the monotony of the interval inclusion used in the interval neuron function proposal (1), this error function E_k depends only on the bounds of the relevant intervals:

$$E_k = \frac{1}{2} \sum_v \left((a_v(I_k) - A_{kv})^2 + (b_v(I_k) - B_{kv})^2 \right) \tag{2}$$

Finally, the total error function E of the multi-layered neural network with respect to the whole training set, is the sum of the partial error function:

$$E = \sum_{k=1}^{N} E_k \tag{3}$$

In the beginning of the learning process, we choose the values of synaptic weights randomly close to zero and gain parameters around one. During the adaptation of the neural network to the given set of interval training patterns, we minimize the error function E (3) in the synaptic weight and gain space (Kufudaki and Hořejš, 1991) using some variant of the gradient method. For example:

$$\begin{aligned} \Delta w_{ij}(t) &= -\varepsilon \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(t-1) \\ \Delta \lambda_i(t) &= -\varepsilon' \frac{\partial E}{\partial \lambda_i} + \alpha' \Delta \lambda_i(t-1) \end{aligned} \tag{4}$$

where $\varepsilon, \varepsilon' > 0$, $0 < \alpha, \alpha' < 1$ are the parameters of the gradient method; w_{ij} is the synaptic weight of the connection leading from neuron j to i; λ_i is the gain parameter of the transfer function S_i of the neuron i, and $\Delta w_{ij}(t)$, $\Delta\lambda_i(t)$ are their relevant increments in the discrete adaptation time t.

To implement the gradient method (4), we need to compute the values of the partial derivations $\frac{\partial E}{\partial w_{ij}}, \frac{\partial E}{\partial \lambda_i}$. For this purpose, we employ the back propagation strategy. After using the rule for composite function derivation, and some calculations, we obtain the resulting formulas (5).

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}} &= \sum_{k=1}^{N} \frac{\partial E_k}{\partial w_{ij}}, \qquad \frac{\partial E}{\partial \lambda_i} = \sum_{k=1}^{N} \frac{\partial E_k}{\partial \lambda_i} \\ \frac{\partial E_k}{\partial w_{ij}} &= \frac{\partial E_k}{\partial a_i} \frac{\partial a_i}{\partial x_i} \frac{\partial x_i}{\partial w_{ij}} + \frac{\partial E_k}{\partial b_i} \frac{\partial b_i}{\partial y_i} \frac{\partial y_i}{\partial w_{ij}} \\ \frac{\partial E_k}{\partial \lambda_i} &= \frac{\partial E_k}{\partial a_i} \frac{\partial a_i}{\partial \lambda_i} + \frac{\partial E_k}{\partial b_i} \frac{\partial b_i}{\partial \lambda_i} \quad \neg FP(i) \end{aligned} \tag{5}$$

where

$$\frac{\partial x_i}{\partial w_{ij}} = \begin{cases} 1 \quad j = 0 \\ a_j s_i(w_{ij})\,(1 + \lambda_i w_{ij} \overline{s}_i(w_{ij})) + b_j \overline{s}_i(w_{ij})\,(1 - \lambda_i w_{ij} s_i(w_{ij})) \quad j \neq 0 \end{cases}$$

$$\frac{\partial y_i}{\partial w_{ij}} = \begin{cases} 1 \quad j = 0 \\ a_j \overline{s}_i(w_{ij})\,(1 - \lambda_i w_{ij} s_i(w_{ij})) + b_j s_i(w_{ij})\,(1 + \lambda_i w_{ij} \overline{s}_i(w_{ij})) \quad j \neq 0 \end{cases}$$

$$\frac{\partial a_i}{\partial x_i} = \begin{cases} 1 & FP(i) \\ \frac{1}{2}\lambda_i\left(1-a_i^2\right) & \neg FP(i) \end{cases} \qquad \frac{\partial b_i}{\partial y_i} = \begin{cases} 1 & FP(i) \\ \frac{1}{2}\lambda_i\left(1-b_i^2\right) & \neg FP(i) \end{cases}$$

$$\begin{aligned}
\frac{\partial a_i}{\partial \lambda_i} &= \frac{\partial S_i}{\partial \lambda_i} + \frac{\partial a_i}{\partial x_i}\frac{\partial x_i}{\partial \lambda_i} = \\
&= \frac{1}{2}\left(1-a_i^2\right)\left(x_i - \lambda_i \sum_{j\in\overline{i}} w_{ij}^2 s_i(w_{ij})\overline{s}_i(w_{ij})(b_j-a_j)\right) \qquad \neg FP(i) \\
\frac{\partial b_i}{\partial \lambda_i} &= \frac{\partial S_i}{\partial \lambda_i} + \frac{\partial b_i}{\partial y_i}\frac{\partial y_i}{\partial \lambda_i} = \\
&= \frac{1}{2}\left(1-b_i^2\right)\left(y_i + \lambda_i \sum_{j\in\overline{i}} w_{ij}^2 s_i(w_{ij})\overline{s}_i(w_{ij})(b_j-a_j)\right) \qquad \neg FP(i).
\end{aligned}$$

For the computation of $\frac{\partial E_k}{\partial a_i}$, $\frac{\partial E_k}{\partial b_i}$ we distinguish two cases:

- Let i be the output layer neuron, then

$$\frac{\partial E_k}{\partial a_i} = a_i(I_k) - A_{ki}, \qquad \frac{\partial E_k}{\partial b_i} = b_i(I_k) - B_{ki}. \tag{6}$$

- Let i be a hidden layer neuron, and let $\underline{i}$ be the set of neurons to which the connection from the output of neuron i leads, and assume that the value of the partial derivations $\frac{\partial E_k}{\partial a_r}$, $\frac{\partial E_r}{\partial b_r}$, $r \in \underline{i}$ have been already computed, then

$$\begin{aligned}
\frac{\partial E_k}{\partial a_i} &= \sum_{r\in\underline{i}}\left(\frac{\partial E_k}{\partial a_r}\frac{\partial a_r}{\partial x_r}\frac{\partial x_r}{\partial a_i} + \frac{\partial E_k}{\partial b_r}\frac{\partial b_r}{\partial y_r}\frac{\partial y_r}{\partial a_i}\right) = \\
&= \sum_{r\in\underline{i},FP(r)} w_{ri}\left(\frac{\partial E_k}{\partial a_r}s_r(w_{ri}) + \frac{\partial E_k}{\partial b_r}\overline{s}_r(w_{ri})\right) + \\
&+ \frac{1}{2}\sum_{r\in\underline{i},\neg FP(r)} \lambda_r w_{ri}\left(\frac{\partial E_k}{\partial a_r}\left(1-a_r^2\right)s_r(w_{ri}) + \frac{\partial E_k}{\partial b_r}\left(1-b_r^2\right)\overline{s}_r(w_{ri})\right) \\
\frac{\partial E_k}{\partial b_i} &= \sum_{r\in\underline{i}}\left(\frac{\partial E_k}{\partial a_r}\frac{\partial a_r}{\partial x_r}\frac{\partial x_r}{\partial b_i} + \frac{\partial E_k}{\partial b_r}\frac{\partial b_r}{\partial y_r}\frac{\partial y_r}{\partial b_i}\right) = \qquad (7) \\
&= \sum_{r\in\underline{i},FP(r)} w_{ri}\left(\frac{\partial E_k}{\partial a_r}\overline{s}_r(w_{ri}) + \frac{\partial E_k}{\partial b_r}s_r(w_{ri})\right) + \\
&+ \frac{1}{2}\sum_{r\in\underline{i},\neg FP(r)} \lambda_r w_{ri}\left(\frac{\partial E_k}{\partial a_r}\left(1-a_r^2\right)\overline{s}_r(w_{ri}) + \frac{\partial E_k}{\partial b_r}\left(1-b_r^2\right)s_r(w_{ri})\right).
\end{aligned}$$

The formulas (6), (7) imply that the distributed computation of the required partial derivations goes gradually from the output layer to the input one, as the name "back propagation" suggests.

3.4 Expert Checking

After the multi-layered neural network has learnt the given training set of interval patterns, it is necessary to check whether the created neural knowledge base is able to infer usable conclusions from it. This can be done by an expert, e.g., by testing reactions to the randomly generated sensible expert inputs, or with the usage of a so-called test set (i.e., a part of the training set that has not been learnt), and through eliminating incorrect network behaviour by additional learning of the correctly formulated patterns in which the network fails. It is useful to iterate this process until the expert is satisfied by the network's ability to infer correct diagnoses. At the end of expert checking of the neural knowledge base, we determine the generalization confidences of the individual expert outputs by statistically measuring the percentage of right output responses.

Because we have chosen the strictly neural approach to expert system design, there is an important issue — how to integrate the implicit knowledge representation in neural network with the explicit representation of known knowledge which is, for example, in the form of rules. Our strictly neural approach does not make the learning of known explicit rules easy. Šíma (1994a) has discussed this problem and proposed a compromise solution based on the derivatives of training patterns that can, as additional information, improve the network's generalization ability in the case when the explicit rule can be expressed by a differentiable function.

4 Inference Engine

The user's view of how the inference engine operates can take the form which follows in our example. The user puts in some (typically not all) values of the inputs, and the neural expert system provides him with the values of outputs with the corresponding confidences:

```
SORE-THROAT = YES, TEMPERATURE = unknown

⟶ ILLNESS = ANGINA                 conf.  = 0.77

  INABILITY-TO-WORK = YES          conf.  = 0.81

  MEDICATIONS = { PENICILLIN }     conf.  = 0.62
```

So, the input for the inference engine of our neural expert system is the vector of the expert input values. Generally, some of them can be unknown. These expert inputs are encoded to the interval values of input neuron states using the coding procedure described in sections 2, 3. This means that the known expert inputs correspond to one-point intervals of the encoded analog values, and the unknown or irrelevant inputs are encoded by full intervals $\langle -1, 1\rangle$. The work of the inference engine is then based on the interval vector function of the multi-layered neural network, computed gradually via layers using the formula (1), that has been described in section 3.2. As a result of this computation, we obtain the interval inner potentials of the output neurons. The next process of the evaluation of the expert outputs and their confidences depends on the manner of their coding.

First let us consider the output neuron i that encodes the expert output using floating-point coding so it holds $FP(i)$. Let this neuron encode the indexes from the interval $\langle p_i, q_i\rangle$ and let $\langle x_i, y_i\rangle$ be its computed interval inner potential. Using the linear transformation (8) we transform this interval to the interval index $\langle g_i, h_i\rangle$.

$$\begin{aligned} g_i &= \frac{q_i - p_i}{2}(x_i + 1) + p_i \\ h_i &= \frac{q_i - p_i}{2}(y_i + 1) + p_i \end{aligned} \tag{8}$$

Let $r_i = \frac{g_i + h_i}{2}$ be the center of this interval. Let us denote by d_i the length of the intersection interval $\langle g_i, h_i\rangle \cap \langle p_i, q_i\rangle$. Then the output v_i of the neuron i with the inference confidence c_i is determined by the formulas (9).

$$v_i = \begin{cases} p_i & r_i < p_i \\ r_i & p_i \leq r_i \leq q_i \\ q_i & q_i < r_i \end{cases} \qquad c_i = 1 - \frac{d_i}{q_i - p_i} \tag{9}$$

By decoding the value v_i in the case of the scalar or set type, we round off the value v_i on the integer.

Conversely, let us have the output neuron i whose state is a part of the binary or unary code of the relevant expert output so it holds $\neg FP(i)$. Let x_i, y_i be the bounds of the computed interval inner potential of the neuron i, and (after eventual renaming) let it be $x_i \leq y_i$. Let $0 < CRT < 1$ determine the so-called critical interval of indecision $(-CRT, CRT)$, where the state of the output neuron does not correspond to any of two possible values -1,1. Using this optional constant CRT we compute the value

$$k_i = |S_i^{-1}(CRT)| = \left| \frac{1}{\lambda_i} \ln \left(\frac{1 - CRT}{1 + CRT} \right) \right| \tag{10}$$

which determines the critical interval of indecision $(-k_i, k_i)$ for the interval inner potential of the neuron i. By comparing $\langle x_i, y_i \rangle$ with $(-k_i, k_i)$ we determine the value of the positive confidence c_i^+ of the output $v_i = 1$, and the value of the negative confidence c_i^- of the output value $v_i = -1$ (11).

$$c_i^+ = \begin{cases} 1 & k_i \leq x_i \\ \frac{y_i - k_i}{y_i - x_i} & x_i < k_i \leq y_i \\ 0 & \text{otherwise} \end{cases} \qquad c_i^- = \begin{cases} 1 & y_i \leq -k_i \\ \frac{-k_i - x_i}{y_i - x_i} & x_i \leq -k_i < y_i \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

We determine the output v_i of the neuron i with the confidence c_i (12) with respect to the dominant confidence.

$$v_i = \begin{cases} 1 & c_i^+ > c_i^- \\ -1 & c_i^+ < c_i^- \\ 0 & \text{otherwise} \end{cases} \qquad c_i = \max\left(c_i^+, c_i^-\right) \tag{12}$$

The value of the inference confidence of the expert output is the result of multiplication of the relevant confidences of its coding neurons. If, by decoding of the expert output, the output of some coding neuron is equal to 0 or the coded value does not correspond to any value in the type definition then this expert output is considered to be unknown.

Finally, the expert outputs are determined by decoding the output values of the coding output neurons. Their inference confidences regarding unknown inputs are multiplied by the generalization confidences, which we acquire from the expert checking of the neural knowledge base mentioned in section 3.4. If the resulting confidence is less than the optional critical confidence (e.g., $\frac{1}{2}$), we consider the value of the relevant expert output to be unknown.

5 Conclusion's Explanation

The explanation heuristics of our neural expert system creates, for the given expert output, the list of the most relevant expert input values and their percentage of influence measurements. It is expected that these input values have the greatest influence on the inferred expert output. In our example, the explanation can take the following form:

```
MEDICATIONS = { PENICILLIN } ?
⟶ SORE-THROAT = YES 100%
```

```
INABILITY-TO-WORK = YES ?
──→ SORE-THROAT = YES 65%
    TEMPERATURE =  unknown 35%
```

The results of the explanation can be used, not only for expert analysis, but it is also possible to debug the neural knowledge base by means of them. For example, if we want to force a rule into the neural network and the system responds incorrectly in some cases with respect to this rule, we can create the appropriate example inferences using the explanation of failed answers. These patterns, after being learnt, regulate the network function correctly. The explanation abilities of our neural expert system are also used for question generating in the interactive mode. The description of the heuristics for the different ways of expert output coding, and of question generating, follows.

5.1 Explanation of Floating Point Coded Output

The heuristics of explanation of the floating-point coded expert output is based on the so-called causal index (Entbutsu, Baba, and Hara, 1991), which is the partial derivation of the neural output by input. This index should measure the degree of the causal relationship between the relevant input and output neurons. In our generalized interval model of the multi-layered neural network, it is possible to consider four types of derivations between the output and input neurons, according to the combinations of the interval bounds. Therefore, the degree v_{ij} of the influence of the input neuron j on the output neuron i ($FP(i)$) is determined by the maximum of the absolute values of these derivations (13).

$$v_{ij} = \max\left(\left|\frac{\partial a_i}{\partial a_j}\right|, \left|\frac{\partial a_i}{\partial b_j}\right|, \left|\frac{\partial b_i}{\partial a_j}\right|, \left|\frac{\partial b_i}{\partial b_j}\right|\right) \tag{13}$$

For $k = 1, \ldots, M$ denote by J_k the set of input neurons which encode the kth expert input, where M is the number of expert inputs. Then the influence V_k of the kth expert input, on the given expert output encoded by neuron i using floating-point, is defined in the following way:

$$V_k = \sum_{j \in J_k} v_{ij} \qquad k = 1, \ldots, M. \tag{14}$$

For all expert inputs the values V_k are computed and ordered so that $V_{k_1} \geq V_{k_2} \geq \cdots \geq V_{k_M}$. Then we determine the minimal index r so that it holds

$$\sum_{s=1}^{r} V_{k_s} \geq \sum_{s=r+1}^{M} V_{k_s}. \tag{15}$$

Put

$$R = \min \left(r, \left\lfloor \frac{M+2}{2} \right\rfloor, 10 \right). \tag{16}$$

The resulting list of expert inputs with percentage of influence measurements (17) that have the greatest influence on the given output is $(k_1, p_1), \ldots, (k_R, p_R)$.

$$p_s = 100 \frac{V_{k_s}}{\sum_{t=1}^{R} V_{k_t}} \qquad s = 1, \ldots, R \tag{17}$$

It remains to describe the computation of the considered partial derivations (13) of an output by inputs. Let i be the output neuron such that it holds $FP(i)$. We again employ the strategy of back propagation:

- Let $j \in \overline{i}$ be the neuron which inputs into neuron i, then

$$\begin{aligned} \frac{\partial a_i}{\partial a_j} &= w_{ij} s_i(w_{ij}), \qquad \frac{\partial a_i}{\partial b_j} = w_{ij} \overline{s}_i(w_{ij}), \\ \frac{\partial b_i}{\partial a_j} &= w_{ij} \overline{s}_i(w_{ij}), \qquad \frac{\partial b_i}{\partial b_j} = w_{ij} s_i(w_{ij}). \end{aligned} \tag{18}$$

- Conversely, let $j \notin \overline{i}$ and assume that the values of the partial derivations $\frac{\partial a_i}{\partial a_k}, \frac{\partial a_i}{\partial b_k}, \frac{\partial b_i}{\partial a_k}, \frac{\partial b_i}{\partial b_k}$, $k \in \underline{j}$ have been already computed, then

$$
\begin{aligned}
\frac{\partial a_i}{\partial a_j} &= \sum_{k\in \underline{j}} \left(\frac{\partial a_i}{\partial a_k}\frac{\partial a_k}{\partial x_k}\frac{\partial x_k}{\partial a_j} + \frac{\partial a_i}{\partial b_k}\frac{\partial b_k}{\partial y_k}\frac{\partial y_k}{\partial a_j} \right) = \qquad (19)\\
&= \frac{1}{2}\sum_{k\in \underline{j}} \lambda_k w_{kj} \left(\frac{\partial a_i}{\partial a_k}\left(1-a_k^2\right) s_k(w_{kj}) + \frac{\partial a_i}{\partial b_k}\left(1-b_k^2\right) \overline{s}_k(w_{kj}) \right),\\
\frac{\partial a_i}{\partial b_j} &= \sum_{k\in \underline{j}} \left(\frac{\partial a_i}{\partial a_k}\frac{\partial a_k}{\partial x_k}\frac{\partial x_k}{\partial b_j} + \frac{\partial a_i}{\partial b_k}\frac{\partial b_k}{\partial y_k}\frac{\partial y_k}{\partial b_j} \right) =\\
&= \frac{1}{2}\sum_{k\in \underline{j}} \lambda_k w_{kj} \left(\frac{\partial a_i}{\partial a_k}\left(1-a_k^2\right) \overline{s}_k(w_{kj}) + \frac{\partial a_i}{\partial b_k}\left(1-b_k^2\right) s_k(w_{kj}) \right),\\
\frac{\partial b_i}{\partial a_j} &= \sum_{k\in \underline{j}} \left(\frac{\partial b_i}{\partial a_k}\frac{\partial a_k}{\partial x_k}\frac{\partial x_k}{\partial a_j} + \frac{\partial b_i}{\partial b_k}\frac{\partial b_k}{\partial y_k}\frac{\partial y_k}{\partial a_j} \right) =\\
&= \frac{1}{2}\sum_{k\in \underline{j}} \lambda_k w_{kj} \left(\frac{\partial b_i}{\partial a_k}\left(1-a_k^2\right) s_k(w_{kj}) + \frac{\partial b_i}{\partial b_k}\left(1-b_k^2\right) \overline{s}_k(w_{kj}) \right),\\
\frac{\partial b_i}{\partial b_j} &= \sum_{k\in \underline{j}} \left(\frac{\partial b_i}{\partial a_k}\frac{\partial a_k}{\partial x_k}\frac{\partial x_k}{\partial b_j} + \frac{\partial b_i}{\partial b_k}\frac{\partial b_k}{\partial y_k}\frac{\partial y_k}{\partial b_j} \right) =\\
&= \frac{1}{2}\sum_{k\in \underline{j}} \lambda_k w_{kj} \left(\frac{\partial b_i}{\partial a_k}\left(1-a_k^2\right) \overline{s}_k(w_{kj}) + \frac{\partial b_i}{\partial b_k}\left(1-b_k^2\right) s_k(w_{kj}) \right).
\end{aligned}
$$

5.2 Explanation of Binary and Unary Coded Output

Now we describe the heuristics for the explanation of the binary and unary coded expert output. The proposal basically keeps the correct philosophy of finding out the dominant influences on the given expert output (Šíma, 1992a).

For better notation of this heuristics, we use an auxiliary data structure, namely, the list of records Z with the following items:

Z.C neuron number (we consider the increasing neuron enumeration from the input layer to the output one)

Z.S influence sign (true,false)

Z.M influence measurement

We assume the decreasing order of the list with respect to $Z.C$, i.e., higher layer neurons are in the beginning of this list. Further, the following operations are implemented for this list:

INSERT(Z) insert the record Z into the list with respect to the ordering

GET(Z) get the first record Z from the list

LIST^ access (buffer) variable to the first record of the list

At the beginning of the procedure, the records for the neurons that encode the given expert output are inserted into the list. The neuron outputs determine the influence signs, and their confidences are the basis for the influence measurements. Then the following general step is repeated. In the case of multiple occurrences of records for one neuron in the beginning of the list, the resulting influence sign and measurement of this neuron is calculated. All the records for this neuron are removed from the list. The neurons with the greatest relevant influence on this neuron are selected, and the new records created for them are inserted into the list. Upon completion of the procedure, only records for the input neurons remain in the list, and with respect to them, the expert inputs with the greatest influence measurements are chosen. Detailed description of the above-mentioned process follows, written in Pascal-type pseudocode:

$Y \longleftarrow$ the set of neurons which encode the given expert output;
$LIST \longleftarrow$ empty list;
forevery $i \in Y$ **do**
  $Z.C \longleftarrow i$;
  $Z.S \longleftarrow (v_i = 1)$; {neuron output}
  $Z.M \longleftarrow c_i$; {output confidence}
enddo;

forevery expert input k **do** $V_k \longleftarrow 0$ **enddo**;
repeat

{the determination of Z for the neuron $LIST\hat{}.C$

which is duplicated in the list}

$M1 \longleftarrow 0$; $M2 \longleftarrow 0$;

repeat

$GET(Z)$;

if $Z.S$ **then** $M1 \longleftarrow M1 + Z.M$ **else** $M2 \longleftarrow M2 + Z.M$ **endif**

until $LIST\hat{}.C \neq Z.C$;

$Z.S \longleftarrow (M1 > M2)$; $Z.M \longleftarrow |M1 - M2|$;

{condition: the list contains the input neurons only}

if $Z.C$ is the input layer neuron **then**

{the creation of V_k, as in section 5.1}

$k \longleftarrow$ expert input coded by the neuron $Z.C$;

$V_k \longleftarrow V_k + Z.M$

else

{the computation of the contributions to the inner

potential of the neuron $Z.C$}

$i \longleftarrow Z.C$;

forevery $j \in \overline{i}$ **do**

if Z.S **then**

$A_j \longleftarrow w_{ij}\,(\overline{s}_i(w_{ij})a_j + s_i(w_{ij})b_j)$

else

$A_j \longleftarrow w_{ij}\,(s_i(w_{ij})a_j + \overline{s}_i(w_{ij})b_j)$

endif

enddo;

{the selection of the contributions with the relevant signs}

$J \longleftarrow \{j \in \bar{i}/\, Z.S = (\lambda_i A_j > 0)\}$;

{the selection of the dominant contributions}

enumerate $J = \{j_1, j_2, .., j_t\}$ so that $|A_{j_1}| \geq |A_{j_2}| \geq \cdots \geq |A_{j_t}|$;

determine minimal u so that $\sum_{s=1}^{u} |A_{j_s}| \geq \sum_{s=u+1}^{t} |A_{j_s}|$;

$J' \longleftarrow \{j_1, j_2, \ldots, j_u\}$;

{the insertion of elements of J' into the list}

forevery $j \in J'$ **do**

$Z1.C \longleftarrow j$;

$Z1.S \longleftarrow Z.S = (\lambda_i w_{ij} > 0)$;

$Z1.M \longleftarrow Z.M\, |A_j|$;

$INSERT(Z1)$

enddo

endif

until $LIST$ = empty list;

We have the values V_k for every expert input available. The resulting list of expert inputs $(k_1, p_1), \ldots, (k_R, p_R)$ with their influence measurements (17), is determined from them in a similar fashion as in section 5.1 (15), so that the number of the influential expert inputs is limited according to the formula

$$R = \min\left(\left\lfloor \frac{M+2}{2} \right\rfloor, 10\right). \tag{20}$$

The limitation of this heuristics, is that it is impossible to give reasons for the unknown expert output, because we cannot determine the influence signs of the neurons which encode it. This corresponds to the situation that the expert does not know some output values and he is not able to explain why he does not know it.

5.3 Generating Questions for Unknown Inputs

The user usually consults particular problems while in the interactive mode, and the neural expert system asks him questions for the unknown values of the expert inputs. The strategy of selection of the appropriate question consists of finding such unknown expert input that, given its value, the confidence and completeness of expert outputs would mostly increase after the next inference. This is not only the matter of one question, but of the whole sequence of questions, because the user can give a response "unknown" to the first question.

Our heuristics of question generating makes use of the explanation described in this section. We gradually give reasons for all known expert output values and add the percentage influence measurements of the unknown inputs. The sequence of questions for the unknown values of expert inputs is then given in decreasing order of these sums. In the case of an empty sequence, i.e., no unknown expert input is the cause of the known expert output values according to the explanation heuristics, we choose an arbitrary sequence of questions for all unknown expert input values.

6 Examples of Application

The above described theoretical results have been used in the software product EXPSYS, which is a tool for neural expert system design. There are three versions of this program and its several successful applications on non-trivial real-world problems.

First version was applied on a 16-25-20-8 net for diagnosis of protection warnings in atomic energy plants (for the cooling system of a VVER 410 V-213 reactor, to be specific) (Šíma, 1992a). Starting from 80 original members of training set, another 32 were subsequently added after consultation with an expert; afterwards, the learning algorithm reached a global error minimum near 0 (after 6 hours on a VAX 3600), and good generalization ability as was proved on a test set of about 300 members, where more than 90% of the answers were accepted (much to his surprise) by an expert, who had been disappointed by previous experience with classical expert systems.

A second version of EXPSYS (Šíma and Neruda, 1993a, 1993b) has been developed as a user-friendly, PC-compatible program with a state-of-the-art interface.

This version has been tested on diagnostics and progress prediction of hereditary muscular diseases - Duchenne's and Becker's dystrophy (Hájek, Grosmanová, Horská, and Urban, 1993). The diagnostic neural expert system for this problem (Šíma and Neruda, 1994) has 16 inputs and 1 output. For the list of them, with their types and coding, see Tab.2. The neural knowledge base (with the topology of a multi-layered neural network 31-15-10-3) was created automatically during 40 minutes (on an IBM PC/AT with an 80386 processor and a mathematical coprocessor) in the process of learning from the set of 63 example inferences. The inference engine has been able, from this knowledge base, to determine a correct output diagnosis from generally incomplete inputs. The explanation heuristics unambiguously pointed out two mostly relevant inputs (EXON,PROGRESE), which according to the expert's opinion, are sufficient for the determination of the resulting output diagnosis (cf. Tab.3).

There are other two applications of the second version of EXPSYS. The first one concerns diagnosis of ear diseases (Mechlová, 1994) using 6 inputs and 2 outputs and with the net topology 6-9-9-9. About 120 example inferences were learnt during 3 days (on IBM PC 486/40MHz) and the system exhibited a performance rate of 95% successful responses on the test set of about 100 examples. Also the correctness of its explanations was confirmed by an expert. The other application deals

Expert inputs		
Name	Type	Coding
EXON	SET OF [4, 8, 12, 17, 19, 44, 45, 48, 51]	BINARY
VEK-ZAC	REAL [0.00; 100.00]	REAL
VEK-VYS	REAL [0.00; 100.00]	REAL
MYOGLOB	REAL [0.00; 2500.00]	REAL
CPK	REAL [0.00; 400.00]	REAL
BIOPS	SET OF [dystrof, atrof, denerv, myopat, steatosa]	BINARY
EMG	SET OF [myog, denspin, denneur]	BINARY
HIP-GIRDLE	SCALAR OF [No, Yes]	BINARY
PROGRESE	SCALAR OF [none, slow, fast]	BINARY
PCR/ATP	REAL [0.00; 2.00]	REAL
PCR/BATP	REAL [0.00; 6.00]	REAL
PCR/PI	REAL [0.00; 7.00]	REAL
PCR/PME	REAL [0.00; 100.00]	REAL
PCR/PDE	REAL [0.00; 10.00]	REAL
PCR/PDE	REAL [0.00; 2.00]	REAL
PDE/BATP	REAL [0.00; 1.00]	REAL
Expert Output		
Name	Type	Coding
FENOTYP	SCALAR OF [normal, dmd, bmd]	UNARY

Table 2: Types and Coding of Expert Inputs and Outputs

with the relationship between the structure and properties of chemotherapeutics for treatment of melanoses (Štrouf, Hálová, Žák, and Kyselka, 1993). The topology 8-10-5 was used here, trained using 35 examples during 30 minutes (on an IBM PC/AT 80386/40MHz). Because of the lack of data, the system was not used for prediction, but reasons given in the explanations were successfully compared with theoretical knowledge and experience.

The general experience with EXPSYS, in all the above-mentioned examples of application, is really very good. Many times, the learning algorithm discovered the hidden inconsistency in a training set, and refused to learn patterns later described by experts as inconsistent or even mistaken. The system responded, in these cases, better than the incorrect patterns recommended. The results of expert checking were first more than 80% correct and the overall quality of inferences was improved after additional learning. Also, explanation heuristics provided relevant reasons for

Inference Inputs		
Name	Value	Influence
EXON	45	62.14
VEK-ZAC	5	1.63
VEK-VYS	12	0.00
MYOGLOB	unknown	0.00
CPK	53.1	0.00
BIOPS	unknown	0.00
EMG	myog	0.00
HIP-GIRDLE	Yes	1.47
PROGRESE	slow	34.76
PCR/ATP	1.15	0.00
PCR/BATP	3.82	0.00
PCR/PI	3.91	0.00
PCR/PME	8.53	0.00
PCR/PDE	unknown	0.00
PI/PDE	1.28	0.00
PDE/BATP	0.74	0.00
Inference Output		
Name	Value	Confidence
FENOTYP	bmd	0.94

Table 3: Example of Inference

inference, in spite of the fact that little information, given only in the set of example inferences, was available.

These surprisingly good results illustrate the great generalization ability of the system, comparable to human intelligence, and confirms the correctness of the heuristics proposals for work with incomplete and irrelevant information, and for explanation. This program, because of its generality, can be applied in arbitrary areas where the middle-complex expert system needs to be created in a relatively short time. We can state that the neural approach to expert system design studied here is a possible alternative to the classical rule-based methods.

Finally, we are working on the third version of EXPSYS, which will be much more user-comfortable than the previous one. It will be also easily adjustable to any other computer environment. The basic configuration will include interfaces for MS Windows and X Windows. We hope that the implementation under the Unix operating system will speed up the learning process, and enable the management of more complicated tasks. We are also exploring the possibility of enriching our system by incorporating explicit rules into the neural knowledge base.

References

Baxt, W.G. (1990). Use of an artificial neural network for data analysis in clinical decision-making: the diagnosis of acute coronary occlusion. Neural Computation, 2, 480-489.

Blum, A.L., Rivest, R.L. (1992). Training a 3-node neural network is NP-Complete, Neural Networks, 5, 117-127.

Caudill, M. (1991). Expert networks, Byte, 16(10), 108-116.

Drucker, H. (1990). Implementation of minimum error expert system, Proceedings of IJCNN, San Diego, I, 291-296.

Enbutsu, I., Baba, K., Hara, N. (1991). Fuzzy rule extraction from a multilayered neural network, Proceedings of IJCNN, Seattle, II, 461-465.

Frasconi, P., Gori, M., Maggini, M., Soda, G. (1991). An unified approach for integrating explicit knowledge and learning by example in recurrent networks, Proceedings of IJCNN, Seattle, I, 811-816.

Gallant, S.I. (1987). Automated generation of connectionist expert systems for problems involving noise and redundancy, AAAI Workshop on Uncertainty in Artificial Intelligence, Seattle, 212-221.

Gallant, S.I. (1988a). Bayesian assessment of a connectionist model for fault detection, AAAI Workshop on Uncertainty in Artificial Intelligence, St. Paul MN.

Gallant, S.I. (1988b). Connectionist expert systems, Communications of the ACM, 31(2), 152-169.

Gallant, S.I. (1993). Neural network learning and expert systems, MIT Press.

Hájek, M., Grosmanová, A., Horská, A., Urban, P. (1993). Comparison of the clinical state and its changes in patients with Duchenne's and Becker's muscular dystrophy with results of in vivo P magnetic resonance spectroscopy, European Radiology, 3, 499-506.

Handelman, D.A., Lane, S.H., Gelfand, J.J. (1989). Integration of knowledge-based system and neural network techniques for autonomous learning machines, Proceedings of IJCNN, Washington, I, 683-688.

Hořejš, J. (1992). Some research topics of a Prague group (a survey), Neural Network World, 2, 661-666.

Hruska, S.I., Kuncicky, D.C., Lacher, R.C. (1991). Hybrid learning in expert networks, Proceedings of IJCNN, Seattle, II, 117-120.

Judd, J.S. (1990). Neural network design and the complexity of learning, MIT Press.

Kasabov, N. (1991). Expert systems based on incorporating rules into neural networks, Visiting lecture at the Dept. of Comp. Sci. Univ. of Essex, Colchester (England).

Krysiak, A. (1991). Application of parallel connectionist model in medical expert system, obtained from author from Inst. of Comp. Sci. of Polish Academy of Sciences.

Kufudaki, O., Hořejš, J. (1991). PAB: Parameters adapting back-propagation, Neural Network World, 1, 267-274.

Mechlová, S. (1994). Applications of neural networks, Diploma thesis (in czech), Czech Technical University, Electrotech. Faculty, Dept. of Control Engineering, Prague.

McMillan, C., Mozer, M.C., Smolensky, P. (1991). Learning explicit rules in a neural network, Proceedings of IJCNN, Seattle, II, 83-88.

Mozer, M.C. (1987). RAMBOT: A connectionist expert system that learns by example, Proceedings of IJCNN, San Diego, II, 693-701.

Rumelhart, D.E., Hinton, G.E., Williams, R.J. (1886). Learning representations by back-propagating errors, Nature, 323, 533-536.

Saito, K., Nakano, R. (1988). Medical diagnostic expert system based on PDP model, Proceedings of IEEE, San Diego, I, 255-261.

Samad, T. (1988). Towards connectionist rule-based systems, Proceedings of IEEE, San Diego, II, 525-532.

Svaytser, H. (1989). Even simple neural nets cannot be trained reliably with a polynomial number of examples, Proceedings of IJCNN, Washington, II, 141-145.

Šíma, J. (1992a). The multi-layered neural network as an adaptive expert system with the ability to work with incomplete information and to provide justification of inference, Neural Network World, 2, 47-58.

Šíma, J. (1992b). Generalized back propagation for interval training patterns, Neural Network World, 2, 167-174.

Šíma, J. (1994a). Generalized back propagation for training pattern derivatives, Neural Network World, 4, 91-98.

Šíma, J. (1994b). Loading deep networks is hard, Neural Computation, 6, 840-848.

Šíma, J. (1994c). Back propagation is not efficient, Technical report V-580, ICS Prague.

Šíma, J., Neruda, R. (1992a). Neural Expert Systems, Proceedings of IJCNN, Beijing.

Šíma, J., Neruda, R. (1992b). Neural networks as expert systems, Neural Network World, 2, 775-784.

Šíma, J., Neruda, R. (1993a). Designing neural expert systems with EXPSYS, Technical report V-563, ICS Prague.

Šíma, J., Neruda, R. (1993b). EXPSYS - a tool for neural expert system design, Proceedings of the NEURONET conference, Prague.

Šíma, J., Neruda, R. (1994). The empty expert system and its application in medicine, Proceedings of European Meeting on Cybernetics and Systems Research, Vienna, 1825-1832.

Štrouf, O., Hálová, J., Žák, P., Kyselka, P. (1993). QSAR of catechol analogs against malignant melanoma by EXPSYS neural expert system, poster at Chemometrics III, Brno.

Yang, Q., Bhargava, V.K. (1990). Building expert systems by a modified perceptron network with rule-transfer algorithms, Proceedings of IJCNN, San Diego, II, 77-82.