Revue Roumaine des ScienceTechnique, Série Électrotechnique et Énergétique

NEURAL NETWORK KNOWLEDGE EXTRACTION

CRISTEA ALEXANDRA., CRISTEA P., OKAMOTO T

513.83: 621.311.161

Abstract: The usage of ANNs in "safety-critical" domains, which include the economic and financial applications, is hindered by their “black box”- type approach, which makes it difficult to verify and debug software that includes ANN components. Significant advantages can be gained by combining the symbolic knowledge of a domain theory (DT), with the empirical sub-symbolic knowledge stored in an ANN trained on examples. Rule extraction adds the needed explanation/comprehension component to the much prized ability of ANN to generalise over a learned set of examples. Compiling rules into the an ANN provides better initial conditions for training the network and can significantly improve the speed of learning. The mixed approach allows building hybrid systems that co-operatively combine ANN and AI techniques, increasing both robustness and flexibility. The paper gives an overview of the bases of ANN knowledge extraction under the form of logical functions. ANN design relations are established.

1. INTRODUCTION

During the last decade considerable effort has been dedicated to "write" and "read" symbolic information into and from artificial neural networks (ANNs) [1, 2, 4, 10-14, 16-17, 22-23, 26, 29, 30, 32, 36-38]. The motivation has been multifold. Primarily, ANNs have shown a very good ability to represent "empirical knowledge", as the one contained in a set of examples, but the information is expressed in a "sub-symbolic" form -- i.e., in the structure, weights and biases of a trained ANN, not directly readable for the human user. Thus, an ANN behaves almost like a "black box", providing no explanation to justify its decisions taken in various instances. This forbids the usage of ANNs in "safety-critical" domains, which include the economic and financial applications, and makes it difficult to verify and debug software that includes ANN components. On the other hand, the extraction of the knowledge contained in an ANN allows the "portability" of the information to other systems, in both symbolic (AI) and sub-symbolic (ANN) forms. There are good reasons to consider that Hybrid Learning (HL) Systems -- able to exploit simultaneously theoretical and empirical data [12-14, 16, 23, 31, 36], would be more efficient than each of the Explanation-Based Learning (EBL) systems - that use only theoretical knowledge in symbolical form, or Empirical Learning (EL) systems -- handling the knowledge in the ostensive/empirical form, working separately. The EBL and EL approaches are complementary in many aspects, so they can mutually offset weaknesses and alleviate inherent problems. It is even hoped that HL Systems could assist the induction of scientific theories, helping discover salient features in the input data, whose importance could otherwise be over-looked. Hybrid systems for economic and financial applications, including stock exchange prediction, benefit from the co-operative use of symbolic rule-based AI modules and sub-symbolic, empirically trained, ANN modules. The basic steps of the mixed learning approach are: (1) compile/encode the available theoretical knowledge (domain theory -- DT) into an adequate ANN, (2) use empirical data -- sets of examples -- to train the network, hence introduce additional knowledge into the ANN, (3) extract the refined theory under symbolic form -- to be used in a classical AI system or to be reinserted into the ANN, the cycle being repeated, until some stopping criteria are satisfied.

2. RULE IMPLEMENTATION

2.1. Canonical Binary Functions

The simplest way to implement discrete rules is in the form of binary logical functions. An n-variable binary logical function f: $\mathbf{B}^n \rightarrow \mathbf{B}$, $n \in \mathbf{N}$, $\mathbf{B} = \{F, T\}$; $y = f(x_1, x_2, ..., x_n)$ attaches one of the truth values F-false or T-true to its output (dependent, consequent) variable y, for each combination of the truth values of the input (independent, antecedent) variables $x_1, x_2, ..., x_n$. Usually, numeric coding is used for the binary logical values *true* (T) and *false* (F): **0** and **1** (unipolar representation), or **-1** and **1** (bipolar representation), respectively. As there are 2^n distinct vectors $x=(x_1, x_2, ..., x_n)$ in the discrete input space $\mathbf{B}^n$ and 2 distinct vectors in the output space, there result 2^{2^n} distinct mappings - logical functions. This fast growing combinatorics is the main cause of the complexity problem of logical function implementation. A binary logical function f can also be seen as the characteristic function of the *truth set* of the function f: $\mathbf{T}_f = \{x \mid x \in \mathbf{B}^n, f(x) = T\}$. This approach readily allows the generalisation to fuzzy logic, in connection with fuzzy set theory. A binary function f, constant with respect to m of its n variables, has the *granularity* $y = 2^{n-m}$. A binary logical function f of granularity one, i.e. constant with respect to all its variables, is either a *tautology*, if $f = T$, or a *contradiction*, if $f = F$. The family of one-variable binary logical functions comprises the *identity* $\boldsymbol{I}(x) = x$, the *negation* $\boldsymbol{N}(x) = \overline{x}$, the *tautology* $\boldsymbol{T}(x) = T$, and the *contradiction* $\boldsymbol{C}(x) = F$. The identity and the negation can be represented as instances of the same symbolical function *f(x)* = x^e, where the exponent *e* is conventionally 1 for identity and -1 for the negation. The *n*-variable *canonical* binary functions have one of the entries in their truth table different of all the other entries. These functions are not constant with respect to any of their variables, so that the granularity is 2^n. Consequently, there are 2^n distinct *conjunctive* (AND, product) canonical binary functions, having a single T entry in the truth table, while all the other entries are F:

$$P_k = \bigwedge_{h=1}^{n} x_h^{e_h} = x_1^{e_1} \wedge x_2^{e_2} \wedge \ldots \wedge x_n^{e_n} = \text{AND}\,(x_1^{e_1}, x_2^{e_2}, \ldots x_n^{e_n}), \qquad (1)$$

$$k = 0,1,\ldots,2^n\text{-}1.$$

Similarly, there are 2^n *disjunctive* (OR, sum*)* canonical binary functions with a single F entry in their truth table and all the other entries equal to T:

$$S_k = \bigvee_{h=1}^{n} x_h^{-e_h} = x_1^{-e_1} \vee x_2^{-e_2} \vee \ldots \vee x_n^{-e_n} = \text{OR}\,(x_1^{-e_1}, x_2^{-e_2}, \ldots x_n^{-e_n}), \qquad (2)$$

$$k = 0,1,\ldots,2^n\text{-}1,$$

Because of their role in Boolean algebra, the conjunction and disjunction are often denoted by product and sum, respectively. The product and sum canonical binary function are related by the De Morgan theorems: $S_k = \overline{P_k}$. Each functions is completely specified by the values of the set of symbolic exponents e_h. The indices of the canonical binary function can be related to the symbolic exponents by the relation: $k = \sum_{h=1}^{n} (e_h + 1) 2^{-h-1}$, which corresponds to writing *k* in base 2.

2.2. Normal Forms of Logical Functions

Any binary logical function can be expressed in the *disjunctive normal form*:

$$f = \bigvee_{k=0}^{2^n-1} (\alpha_k \wedge P_k) \qquad (3)$$

or in the *conjunctive normal form*:

$$f = \bigwedge_{k=0}^{2^n-1} (\alpha_k \wedge S_k) \qquad (4)$$

where the characteristic coefficients $\alpha_k \in$ B; $k = 0,\ldots,2^n$ uniquely define the function. The expansion of a logical function in one of the normal forms actually provides the highest granulation analysis of its truth table. As well known, Veitch diagrams - in which the locations correspond to the canonical product or sum functions (components) while the entries are the characteristic coefficients - can be used to minimise the expansions of the functions, by combining the terms which differ only in the values of one variable.

2.3. ANN Implementation of Normal Forms

Fig. 1a presents the standard structure of a first order neuron, while Fig. 1b gives the simplified flow-graph representation of the neuron.

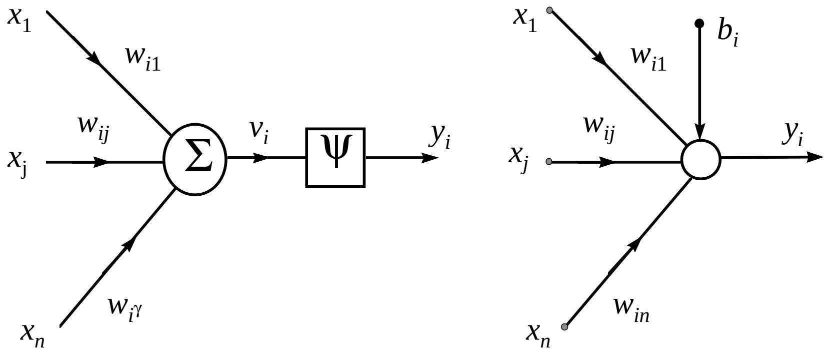


Fig. 1. (a) Standard structure of a first order neuron, (b) Simplified flow-graph representation

The internal activation potential v_i of neuron i is an affine function of its inputs:

$$v_i = \sum_{j=1}^{n} w_{ij}\, x_j + b_i \tag{5}$$

where b_i is the bias (sometimes, the threshold $\theta_i = -b_i$ is used), and w_{ij} are the weights of the links from the input x_j , j=1,...,n, to the neuron i.

The (external) activation y_i of the neuron (the output) is given by:

$$y_i = \psi(v_i) \tag{6}$$

where $\psi(v_i)$ is the activation function of the neuron, usually of the sigmoid type, either unipolar (7, Fig. 2.a) or bipolar (8, Fig 2.b.), depending on the chosen representation of the truth values. The slope parameter of the sigmoid is denoted by σ_i.

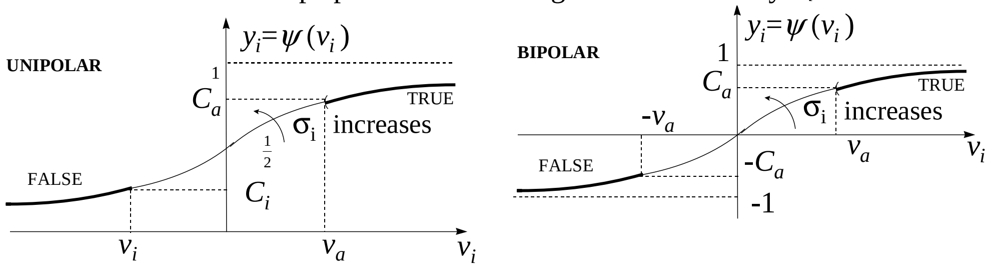


Fig. 2.(a) Unipolar sigmoid activation function (b) Bipolar sigmoid activation function

$$y_i = \psi(v_i) = \frac{1}{1+\exp(-\sigma_i v_i)} \tag{7}$$

$$y_i = \psi(v_i) = \tanh\frac{\sigma_i v_i}{2} \tag{8}$$

The correspondence between the output truth value o_i of a neuron, and the output activation y_i, and the internal activation potential v_i corresponding intervals is given in the table 1.

Table 1

o_i	UNIPOLAR y_i	UNIPOLAR v_i	BIPOLAR y_i	BIPOLAR v_i
true	$(C_a, 1)$	(V_a, ∞)	$(C_a, 1)$	(V_a, ∞)
false	$(0, C_i)$	$(-\infty, V_i)$	$(-1, -C_a)$	$(-\infty, -V_a)$

For the unipolar case: $V_a = \dfrac{1}{\sigma_i}\ln\dfrac{C_a}{1-C_a}, V_i = \dfrac{1}{\sigma_i}\ln\dfrac{C_i}{1-C_i}$ (9).

Typically, one chooses: $0 < C_i < \dfrac{1}{2} < C_a < 1$ and $C_a + C_i = 1$, which results in $V_a > 0$, $V_i = -V_a$. A usual numeric choice is [36]: $C_i = 0.9$, $\sigma_i \in [1,4]$.

For the bipolar case: $V_a = \dfrac{1}{\sigma_i}\ln\dfrac{1+C_a}{1-C_a}$. (10)

To establish the state of a unit implementing a rule as a binary logical function, let us denote by P^+ (P^-) the set of satisfied (unsatisfied) positive antecedents (inputs) and by N^+ (N^-) the set of satisfied (unsatisfied) negative antecedents of the rule. The activation levels of the inputs are shown in table 2. The cardinals of these sets are denoted by the corresponding small letters, while $p = p^+ + p^-$ and $n = n^+ + n^-$ are the total positive and negative antecedents, respectively, and $M = p^+ + n^+$ -- the number of satisfied antecedents, out of the total of $N = p + n$ antecedents.

Table 2

			x_j	
input	w_{ij}	Truth value	Unipolar	Bipolar
$j \in P^+$	> 0	*true*	$(C_a, 1)$	$(C_a, 1)$
$j \in P^-$	> 0	*false*	$(0, C_i)$	$(-1, -C_a)$
$j \in N^+$	< 0	*false*	$(0, C_i)$	$(-1, -C_a)$
$j \in N^-$	< 0	*true*	$(C_a, 1)$	$(C_a, 1)$

Let us also denote by $S_w(Q) = \sum_{j \in Q} |w_{ij}|$ the sum of the absolute values of the weights from the inputs in the set Q. For the unipolar implementation, the activation v_i of the unit i, for a given arbitrary (logical) input vector, falls between the bonds:

$$v_{i\ \min} = C_a\ S_w(P^+) - C_i\, S_w(N^+) + 0 \cdot S_w(P^-) - S_w(N^-) + b_i\,,$$

$$v_{i\ \max} = S_w(P^+) - 0 \cdot S_w(N^+) + C_i \cdot S_w(P^-) - C_a\, S_w(N^-) + b_i\,. \quad (11)$$

When implementing rules as logical functions, the same absolute value w is assigned to all the weights of the links from the related inputs, so that:

$$v_{i\ \min} = C_a\ p^+ w - C_i\ n^+ w - n^- w + b_i\,,$$

$$v_{i\ \max} = p^+ w - C_i \cdot p^- w - C_a\, n^- w + b_i\,. \quad (12)$$

The output of the unit is certainly *true* and, respectively, *false* for the border conditions: $v_{i\,min} > V_a$, $v_{i\,max} < V_i = -V_a$. (13)

In the ideal case ($C_a = 1$, $C_i = 0$, $v_a=v_i=0$), it results:

$$v_{i\ \min} = v_{i\ \max} = S_w(P^+) - S_w(N^-) + b_i = (p^+ - n^-)w + b_i\,. \tag{14}$$

The relations for the bipolar implementation are:

$$v_{i\ \min} = C_a\left[S_w(P^+) - S_w(N^+)\right] - \left[S_w(P^-) + S_w(N^-)\right] + b_i$$

$$v_{i\ \max} = S_w(P^+) + S_w(N^+) - C_a \cdot \left[S_w(P^-) + S_w(N^-)\right] + b_i \tag{15}$$

and

$$v_{i\ \min} = C_a\ M\ w - (N - M)w + b_i \tag{16}$$

$$v_{i\ \max} = M w - C_a(N - M)w + b_i$$

which give in the ideal case

$$v_{i\ min} = v_{i\ max} = (2M-N)w+b_i. \tag{17}$$

Various symmetrical *at-least-M-of-N* logical functions can be implemented by Choosing different values of the bias b_i (see section 2.5). The unipolar and bipolar implementation of conjunctive and disjunctive canonical functions (AND, OR) are shown in the figures 3 and 4, respectively.

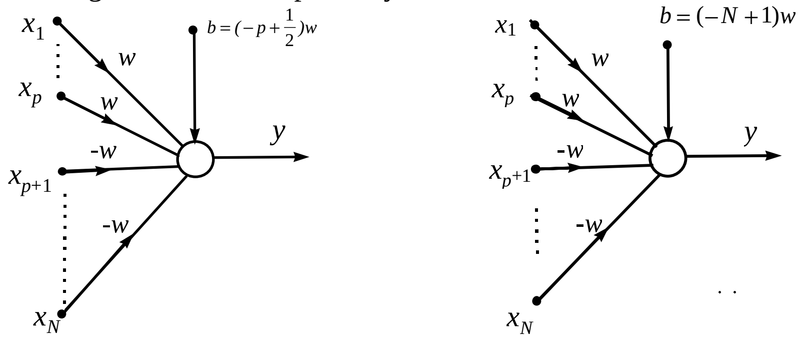
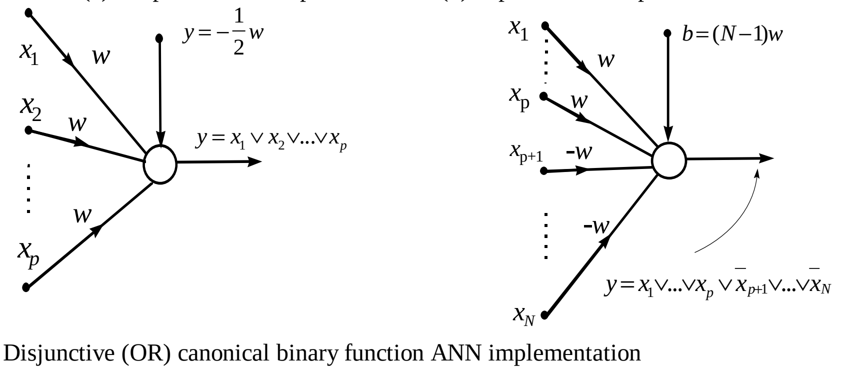


Fig. 3. Conjunctive (AND) canonical binary function: $y = x_1 \wedge x_2 \wedge \ldots \wedge x_p \wedge \overline{x_{p+1}} \wedge \overline{x_{p+1}} \wedge \ldots \wedge \overline{x_N}$

(a) Unipolar ANN implementation, (b) Bipolar ANN implementation



x_1
x_2
x_p
w
w
w
$y=-\frac{1}{2}w$
$y = x_1 \vee x_2 \vee \ldots \vee x_p$
x_1
x_p
x_{p+1}
x_N
w
w
$-w$
$-w$
$b=(N-1)w$
$y = x_1 \vee \ldots \vee x_p \vee \overline{x}_{p+1} \vee \ldots \vee \overline{x}_N$

Fig. 4. Disjunctive (OR) canonical binary function ANN implementation

(a) Unipolar (b) Bipolar

The unipolar implementation of the disjunctive canonical binary function does not allow the negation of the inputs inside the unit. When necessary, the negation of some inputs must be performed in an additional step, preceding the OR function. Figure 5 presents the ideal *true-false* separation planes in the three input variables case, for biases that generate the canonical symmetrical functions from OR (1-of-3) to AND (3-of-3), in both the unipolar and the bipolar implementation.

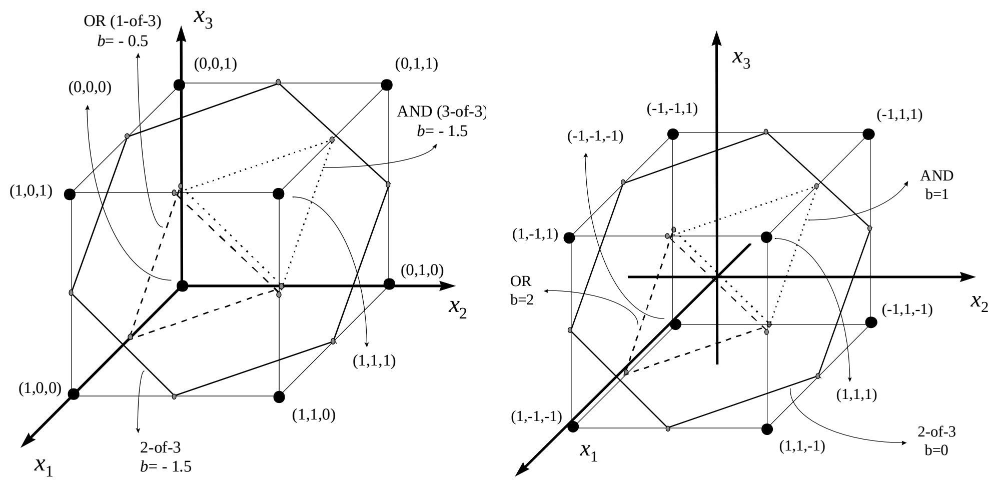


Fig. 5. Input space for equal weights ANN implementation of logical functions
(a) Unipolar (b) Bipolar

The ANN implementation of the disjunctive normal form of a logical function is shown in figure 6 for the case of four variable logical functions.

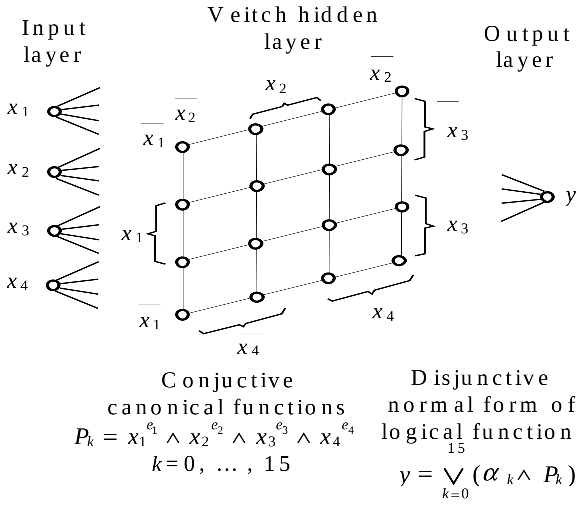


Fig. 6. ANN implementation of the disjunctive normal form of a logical function

The network comprises an input layer with the four variables, a hidden layer with nodes corresponding to the conjunctive canonical functions (components) disposed in a Veitch like structure and an output layer containing the logical function to be implemented. The training of the network consists, in this case, only in the selection of the conjunctive canonical functions which contribute in the OR function of the output unit, while the links between the first two layers are kept frozen. After the training, the unnecessary nodes of the hidden layer are pruned The neighbouring nodes with similar outputs can be combined in a procedure inspired from the logical function minimisation. This structure is adequate for extracting rules with the SUBSET algorithm (3.2), but has the same intrinsic complexity problem.

2.4. Shannon's Symmetrical Logical Functions

Symmetrical functions have the property that any permutation of the variables does not change the output value of the function (e.g., $f_a = x_1\, x_2 + x_2\, x_3 + x_3\, x_1$, $f_b = x_1 \overline{x_2}\, x_3 + \overline{x_1}\, x_2 x_3 + \overline{x_1}\, \overline{x_2}\, \overline{x_3}$). Shannon numbers theorem states that any symmetrical function is fully defined by a set of numbers $\{\beta_1, \beta_2, \ldots \beta_k\}$, so that if exactly β_j, $j=1,2, \ldots, k$ variables are *true* (or *false*), the function is *true* (or *false*). For instance, f_a has the β numbers 2 and 3, while f_b has $\beta = 1$. As β can take values form 0 to N - the total number of variables, there are 2^{N+1} distinct symmetrical functions, out of which N+1 are *exactly-M-of-N* symmetrical canonical functions (i.e., that have only one β number). If a symmetrical function contains one term of a canonical symmetrical function, it must contain all the terms of this canonical symmetrical function. A symmetrical function can be decomposed as a disjunction (or a conjunction) of canonical symmetrical functions u_k (v_k), each written in conjunctive (disjunctive) form, similarly to the normal forms expansions (3) and (4):

$$f = \bigvee_{k=0}^{N} (\alpha_k \wedge u_k) = \bigwedge_{k=0}^{N} (\alpha_k \vee v_k)\,, \qquad (18)$$

where the characteristic coefficients $\alpha_k \in B$; $k = 0,\ldots,N$, uniquely define the symmetrical logical function.

In the ANN approach, functions of the type *at-least-M-of-N* are implemented directly. Such functions can easily be expressed in terms of the *exactly-M-of-N* canonical symmetrical functions:

$$U_j = u_0 + u_1 + \ldots + u_j\,; \qquad j = 0,1,\ldots,n. \qquad (19)$$

The inverse relation is also immediate:

$$u_0 = U_0,\ u_j = \overline{U}_{j-1}\, U_j, \quad j = 1,2,\ldots,n \qquad . \qquad (20)$$

The canonical symmetrical functions have the properties:

$$u_i\, u_j = \mathrm{F},\ v_i + v_j = \mathrm{T}; \qquad \text{for } i \neq j \qquad (21)$$

2.5. ANN IMPLEMENTATION OF SYMMETRICAL LOGICAL FUNCTIONS

Most of the real world problems present symmetries with respect to some groups of variables. For every such group symmetrical functions can be implemented much simpler that arbitrary logical functions [361]. The *at-least-M-of-N* functions are implemented with the same weights as the AND (*N-of-N*) and OR (1-*of-N*) functions (figures 3 and 4), changing only the bias to $b = (-M + 1/2)w$ in the unipolar case, and $b = (N - 2M + 1)w$ – in the bipolar case (see 2.3). The u_j "*exactly-M-of-N*" functions are conjunctions of two U_j functions (20). Figure 7 presents the structure of an ANN implementing a symmetrical logical function with four variables. When training such a network, only the links to the output node are modified according to the examples that define the considered logical function.

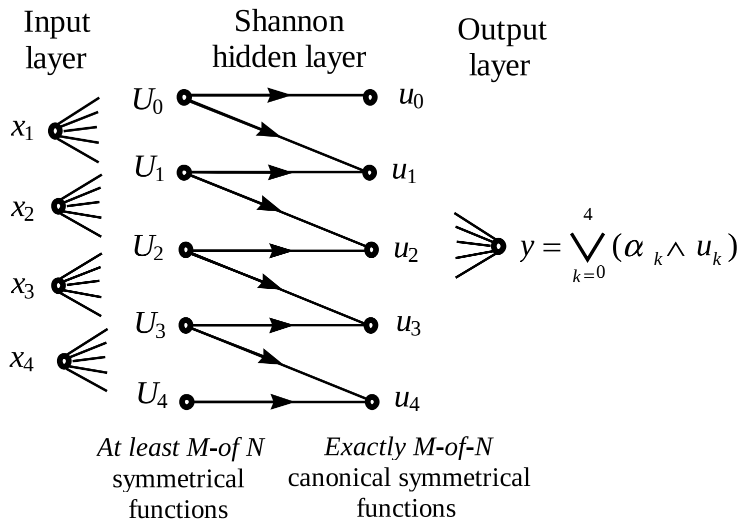

Input layer
Shannon hidden layer
Output layer
x_1
x_2
x_3
x_4
U_0
U_1
U_2
U_3
U_4
u_0
u_1
u_2
u_3
u_4
$y = \bigvee_{k=0}^{4} (\alpha_k \wedge u_k)$
At least M-of N symmetrical functions
Exactly M-of-N canonical symmetrical functions

Fig. 7. Symmetrical functions ANN implementation by decomposition in canonical symmetrical functions

The resulting network has an architecture adequate for the *M-of-N* algorithm (3.3).

3. RULE EXTRACTION ALGORITHMS

3.1 Classification of rule extraction algorithms

There are two main approaches to knowledge extraction: (1) *decompositional* (structural analysis) methods - which assign to each unit of the ANN a prepositional variable, and establish the logical links between these variables; (2) *pedagogical* (input-output mapping) methods, which treat the network as a black-box, without analysing its internal structure. From the algorithms in the literature, *KBANN*, *KT*, *Connectionist Scientist Game*, *RULE NEG*, *RULE-OUT* are based on the decompositional approach, while *BRAINNE* illustrates the pedagogical methods. This paper is focused on the decompositional approach, which puts in correspondence the ANN *sub-symbolic* structure with the theory *symbolic* structure.

3.2 SUBSET Algorithm

The method operates based on the relation (14) giving the ideal *true-false* border of an unipolar neural unit of an ANN: $v_{i_{\text{border}}} = S_w(P^+) - S_w(N^-) + b_i$. This border relation is based on the realistic assumption that the levels of the input/output signals of neurons in a properly trained network, storing rules as logic functions, correspond to either *true*, or *false*, so that the activation potential is determined by the values of the weights. By finding subsets of the incoming links for which the sum of the weights, plus the bias, makes the internal activation potential high enough to bring the output variable at the *true* level, it is possible to formulate rules of the form: **if *(condition)* then *(proposition)***, where ***(condition)*** expresses the border relation in terms of its logic meaning, and ***(proposition)*** is the statement attached to the examined neuron. The main difficulty of the SUBSET algorithm consists in its combinatorial complexity, which results in an exceedingly large number of subsets and rules, many redundant. One way to alleviate the problem is setting an upper limit on the number of considered subsets.

*The steps of the **SUBSET** algorithm* are (using the notations defined in 2.3):

1.	For each output and hidden neuron i Find up to $g_p < 2^P$ subsets $\mathsf{P} \in G_p \subset 2^P$ of positive weight links incoming to neuron i, so that: $S_w(\mathsf{P}) + b_i > 0$
2.	For each $\mathsf{P} \in G_p$ Find up to $g_n < 2^n$ subsets $\mathsf{N} \in G_n \subset 2^N$ of negative weight links incoming to neuron i, so that: $S_w(\mathsf{P}) - S_w(N) + b_i > 0$
3.	For each $\mathsf{N} \in G_n$ State de rule: "**if** P **and** N **then** ***(statement attached to unit i)***".
4.	Remove the duplicate rules from the maximum allowed $g_p g_n$ rules for unit i.

This algorithm is currently used by many rule eliciting systems, but -- even with the restriction imposed by g_p and g_n, which can adversely affect accuracy, it generates complicated sets of rules.

Possible improvements are: - keep frozen the parst of the network that correspond to the initial DT, and extract only the additional rules resulted from EL;

- construct hierarchical rules, using logical function minimisation techniques, - start with predefined Veitch hidden layers, and use training only to select connected units.

3.3. *M-of-N* Algorithm

In real world problems, it is common to have sets of variables playing equivalent roles in the analysed system, so that there is an inherent symmetry of the rules to be extracted. This symmetry can greatly reduce the complexity of the extraction procedure. Moreover, imposing from start the symmetry restrictions predicted by the DT, simpler ANNs and learning algorithms result. The *M-of-N* method is based on the idealised relation (17): $v_{i_{\text{border}}} = (2M - N)w + b_i$, where the bias is chosen $b_i = (N - 2\,M_{\text{border}} + 1)\,w$, so that:

$$v_{i_{\text{border}}} = \left[2\left(M - M_{\text{border}}\right) + 1\right]w = \begin{cases} > 0, & \text{when} \quad M > M_{\text{border}} \\ < 0, & \text{when} \quad M < M_{\text{border}} \end{cases} \tag{22}$$

*The steps of the **M-of-N** algorithm are:*

```
1.  For each output and hidden neuron i
        Form groups (clusters) of similarly weighted links
            Replace the individual weights of all the links in a group with
            the average value for the group.
    (This step is not necessary if symmetry is imposed from the start of the training,
    using a modified BP).
2.  Eliminate insignificant groups (that can not change the true/false output state).
3.  Keeping the weights constant, use BP to readjust the biases
4.  For each neuron considered at (1)
        Construct a single rule of the form
            if (at-least-$M_{border}$-of-N antecedents are satisfied)
                then (statement attached to unit  i)
5.  Whenever possible, combine rules to reduce the overall number.
```

Clustering of the links can efficiently be done by specialised algorithms, but if the existing DT gives the necessary hints, it is simpler to impose symmetry in the BP algorithm.

4. CONCLUSIONS

The paper presents an overview of the most frequently used knowledge-as-rules extraction algorithms and provides some design relations that give a better insight on the necessary conditions for an ANN to properly learn logical functions. The methods can be extended for continuously varying variables, by using *the Locally Responsive Units* approach. For certain problems and types of knowledge, when computing the internal activation potential, the additive aggregated input can be replaced with an multiplicative aggregated input, resulting in *second order units* which can better represent the knowledge [29]. When implementing fuzzy logical functions, it can be advantageous to use a *min-max* operator to evaluate the neuron's (internal) activation potential, instead of the classical *affine* operator (5). An efficient training of the ANN must be able to change both the weights of the links and the structure of the network. The incremental approach has to be balanced with a step of pruning, to keep complexity as low as possible for the given problem. Maintaining constant some parts of the network during the training, while changing others, allows the refinement of the DT by finding only the additional rules [12]. This is to be preferred, whenever possible, to the changing of most/all the rules, which leads to loosing the meanings of the initial concepts in the DT.

The implementation of symbolic AI rules in ANN form, and the rule extraction from the ANN sub-symbolic form are the two sides of a potentially powerful tool that jointly uses the theoretical and the empirical knowledge in solving real world problems. This approach seems to have significant advantages for the Stock Exchange Prediction problem.

Received October 10, 1997

** University of Electro-Communications, Tokyo*
Graduate School of Information Systems
***"Politehnica" University of Bucharest,*
Electrical Engineering Department

REFERENCES

1. Andrews, R., Diederich ,J., Tickle, A., "A survey and critique of techniques for extracting rules from trained ANN", *Neurocomputing Research Centre, Queensland University of Technology, Brisbane,* 1995, on www.qut.edu.au
2. Andrews, R. and Geva, S., "Rule extraction from a constrained error back propagation MLP", *Proc. 5th Australian Conference on Neural Networks,* Brisbane Queensland ,1994, pp. 9-12.
3. Ankenbrand, T. and Tomassini, M., "Multivariate time series modeling of financial markets with artificial neural networks", in *Artificial Neural Nets and Genetic Algorithms*, Springer Verlag,Wien, 1995, pages 257-260.
4. Craven, MW, Shavlik, JW, "Using Sampling and Queries to Extract Rules from trained neural networks", *Machine Learning, Proc. of. the Eleventh Int. Conf.*, San Francisco, CA, 1994.
5. Cristea, Alexandra and Okamoto, T., "Neural Networks for Time-Series prediction; Stock Exchange Forecasting", in *Proceedings of the National Conference of IPSJ,* Chiba, Japan, 3/12-14, 1997}, *Information Processing Society of Japan,* vol 2, pages 2-309/310, 1997.
6. Cristea, Alexandra and Okamoto, T., "Neural Networks for Time-Series prediction; Application Field: Stock Exchange", in *Proceedings of the National Conference of IEICE, Artificial Intelligence*, Osaka, Japan, 3/24-26,1997}, The Institute of Electronics, Information and Communication Engineers, Symposium Honorific Session, vol. 1, pages SD1-6, 1997.

7. Cristea, Alexandra and Okamoto, T., "Stock Exchange Analysis as Time-Series; Forecasting with Neural Nets", in *Proceedings of the National IEICE Meeting on Artificial Intelligence,* Chiba, Japan, 3/18, 1997}, The Institute of Electronics, Information and Communication Engineers, pages 9-16, 1997.
8. Cristea, Alexandra and Okamoto, T., "Stock Exchange Forecasting with the Help of Neural Networks", in *Proceedings of the National Conference of IPSJ, Artificial Intelligence*, Osaka, Japan, 9/4-6, 1996, *Information Processing Society of Japan*, vol.2, pages 2-201/2.
9. Cristea, P., "Neural Networks as Tools for Knowledge Eliciting", in *Proceedings of NEUREL'97*, the *4th Seminar on NN Applications in Electrical Engineering*, Sept. 8-9, 1997, Belgrade, pp.2-9.
10. Decloedt, L., "Explicitation de connaisances dans un systeme hybride d'intelligence artificielle", *Computer Science DEA Research Report*, LIFIA-IMAG, Grenoble, France, 1995
11. Demartines, P., Herault, J., "Curvilinear Component Analysis: A self-Organizing Neural Network for Nonlinear Mapping of data Sets", *IEEE Trans. on Neural Networks*, vol. 8, no. 1, January 1997
12. Fu, L.M., "Knowledge-based connectionism for revising domain theories" *IEEE Transactions on Systems, Man and Cybernetics,* vol. 23, no. 1, pp 173-182, 1993..
13. Fu, L.M., "Rule generation from neural networks", *IEEE Transactions on Systems , Man and Cybernetics*, vol. 28, no. 8, pp. 1114-1124, 1994
14. Gallant, S., "Connectionist expert systems", *Communications from the ACM,* vol. 31, no. 2, pp 152-169, 14, February 1988
15. Gas, B. and Natowicz, R., "Unsupervised learning of temporal sequences by neural networks, in Artificial Neural Nets and Genetic Algorithms", Springer Verlag,Wien, 1995, pages 253-256.
16. Hayashi, Y., "A neural expert system with automated extraction of fuzzy if-then rules", in *Advances in Neural Information Processing Systems*, Denver, 1990, vol. 3, pp 578-584, CO. Morgan Kaufmann.
17. Healy, M., Caudell, T, "Acquiring Rule Sets as a Product of Learning in a Logical Neural Architecture", *IEEE Trans. on Neural Networks*, vol. 8, no. 3, May 1997.
18. Izzo, G., Pepicelli, G., Tocchetti, G., "A backpropagation neural network for the study of prediction in management systems", in *N.N.WIRN VIETRI-92, Sixth Italian Workshop,* Ed. E.R. Caianiello, Word Scientific,1992, pages 386-393.
19. Komo, D., Chang, C.I. and Ko, H.,."Neural Network Technology for Stock Market Index Prediction", in *International Symposium on Speech, Image Processing and Neural Networks,* 13-16 April 1994, Hong-Kong, IEEE, pages 543-546.
20. Krogh, A. and Hertz, J.A., "A Simple Weight Decay Can Improve Generalization", Niels Bohr Institute, Copenhagen, Denmark, Computer and Information Sciences, Univ. of California Santa Cruz,CA 95064, 1995.
21. Levin, A.U., Leen, T.K., Moody, J.E., "Fast Pruning Using Principal Components", *in Advances in Neural Information Processing* , J.Cowan, G.Tesauro, J.Alspector, eds., Morgan Kaufmann, San Mateo, CA, 1994.
22. Masuoka, R., Watanabe, N., Kawamura, A., Owada, Y., and Asakawa, K. (). "Neurofuzzy systems fuzzy inference using a structured neural network",. in *Proceedings of the International Conference on Fuzzy Logic & Neural Networks*, 1990, pp. 173-177
23. McMillan, C., Mozer, M. C., and Smolensky, P., "The connectionist scientist game: Rule extraction and refinement in a neural network", in *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society,* Chicago, IL, 1991
24. Meunier, C. and Nadal, J.P., "Sparsely Coded Neural Networks", in *The Handbook of Brain Theory and Neural Networks,* ed. Arbib M.A., MIT press, 1995, pages 899-901.

25. Miyano, T. and Girosi, F., "Forecasting Global Temperature Variations by Neural Networks", Massachusetts Institute of Technology AI Lab. and Center for Biological and Computational Learning, Dept. of Brain and Cognitive Sciences, ftp: publications.ai.mit.edu., 1995.
26. Moody, J. "Prediction Risk and Architecture Selection for Neural Networks", *NATO ASI Series F*, Springer-Verlag, 1994.
27. Murata, N., Yoshizawa, S. and Amari, S., "Network Information Criterion - Determining the Number of Hidden Units for an Artificial NN Model", Dept. of Mathematical Eng. and Info. Physics, Fac. of Engineering, Univ. of Tokyo,ftp-source, 1995.
28. Nelson, M.E., Furmanski, W., Bower, J.M., "Simulating Neurons and Networks on Parallel Computers", in *Methods in Neuronal Modeling, From Synapses to Networks*, ed. C.Koch and I.Segev, MIT, 1989, chapt. 12.
29. Omlin, C., Giles, L., "Extraction of Rules from Discrete-time Recurrent Neural Networks", *Neural Networks* , vol. 9, no. 1, pp. 41-52, 1996
30. Pratt, L. Y. and Kamm, C. A., "Direct transfer of learned information among neural networks", in *Proceedings of the Ninths National Conference on Artificial Intelligence* Anaheim, CA, 1991, pp. 584-589
31. Sestito, S. and Dillon, T., "Using multi-layered neural networks for learning symbolic knowledge", in *Proceedings of the 1990 Australian Artificial Intelligence Conference*, Perth, Australia, 1990
32. Shavlik, J. W., Mooney, R. J., and Towell, G. G., "Symbolic and neural net learning algorithms: An empirical comparison". *Machine Learning*, no. 6, pp. 111-1 43, 1991
33. Shawn, P.D., Davenport, M.R., "Continuous-Time Temporal Back-Propagation with Adaptable Time Delays", in *IEEE Transaction on Neural Networks*, April, Canada, 1992.
34. Swingler, K., "Prediction, Some Pointers, Pitfalls, and Common Errors", Center of Cognitive and Computational Neuroscience, Stirling Univ., Stirling, FK9 4LA, ftp source, 1994.
35. Thimm, G., Fiesler, E., "Neural Network Pruning and Pruning Parameters", 1st OWSC, http://www/bioele.nuee.nagoya-u.ac.jp/wsc1, *1st Online Workshop on Soft Computing*, Nagoya, Japan, 1996.
36. Towell, G. G. and Shavlik, J. W., "The extraction of refined rules from knowledge based neural networks", *Machine learning*, vol. 131, pp. 71-101, 1993.
37. Tresp, V., Hollaty, Ahmad, S., "Network Structuring Using Rule Based Knowledge", in *Neural Information Processing Systems*, 1993, pp.871-878.
38. Weigand, A. S., Rumelhart, D. E., and Huberman, B. A., "Generalization by weight-elimination with application to forecasting", in *Advances in Neural Information Processing Systems*, Denver, 1990, vol. 3, pp. 875-882