Backpropagation for Neural DNF– and CNF–Networks

Christoph Herrmann
FG Intellektik, TH Darmstadt, Germany
64283 Darmstadt, Alexanderstr. 10
chris@intellektik.informatik.th-darmstadt.de

Andreas Thier
athier@intellektik.informatik.th-darmstadt.de

Abstract

The architecture of a neural network with its links and weights can be viewed as a knowledge representation. To overcome the *black–box problem* of not knowing what knowledge is hidden in a neural architecture, we present a strictly logically operating network. Each neuron represents either a disjunction or conjunction of its inputs and the net thus performs the function of a logic formula. This formula can be extracted from the net after completed training. The well–known backpropagation algorithm is adapted to train such logical neural nets. An on–line pruning algorithm is also implemented to eliminate redundant weights.

1 Introduction

Recently many approaches deal with the extraction of knowledge from trained neural networks to overcome the *black–box phenomenon* [1]. Due to the many degrees of freedom of neural processing units, many of the applied techniques bear the difficulty of interpreting the function of the networks. It has been shown that neurons are able to perform logical operations on their inputs ([7], S. 429ff.). We define certain properties for neurons to identify them as conjunctive or disjunctive neurons. Thus, we restrict the degrees of freedom for each neuron and lose some of the network's generalization ability but we gain a well defined operation within the net. Hence, we can extract logical rules from the trained network. Since the standard backpropagation algorithm would adjust the weights inside the network such that the conjunction and disjunction criteria are no longer met, we had to adapt the learning algorithm. Since it is important for the clarity of the extracted rule base to contain a small number of rules of low complexity we eliminated weights that did not contribute significantly to the network's performance. This resulted in an on–line pruning algorithm similar to the one in [3].

2 Logical neurons

2.1 Neural representation of rules

The syntactical structure of logical formulae is based on atoms and operators. An atom being an unconnected proposition with a defined truth value. A literal l is either an atom a or its negation $\neg a$. Formulae are created by connecting literals via operators such as $l_1 \wedge l_2$ or $l_1 \vee l_2$ as well as any arbitrary combination of these.

For the purpose of interpreting neural networks as a representation of logical formulae the activation of the input neurons must represent the truth values of a formula's literals and the activation of the output neuron must be equal to the truth value a whole formula. We will not consider binary *true* and *false* values but analog values that represent the degree of truth to propositions as *A is true* and *A is false*. As in fuzzy logic [10], we will use membership functions μ_A respectively $\mu_{\neg A}$ as a notation. The μ-values will be assigned to the input neuron's activation.

Definition 2.1 A neural network *represents a fuzzy-logical formula* if and only if the activations of its input neurons are equal to the membership values of the formula's fuzzy variables and the activation of its output neuron is equal to the truth value of the represented formula. ■

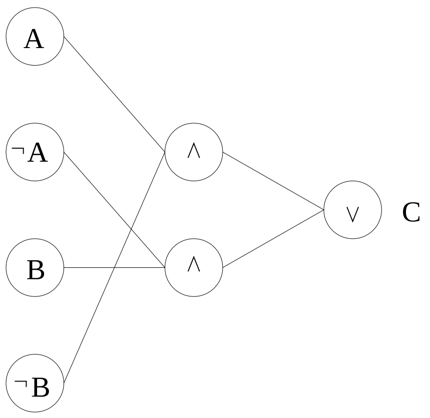

Figure 1: Logically interpretable neural representation of XOR

Figure 1 depicts the neural architecture for a logically interpretable XOR. The implication $A \oplus B \rightarrow C$ is equivalent to the disjunction of conjuntions $(A \wedge \neg B) \vee (\neg A \wedge B) \rightarrow C$ that is performed by the net.

2.2 The architecture

Definition 2.2 The regarded *fuzzy-logically interpretable* neural networks are feed-forward multilayer-perceptrons and have the following properties:

1. $\mathcal{N} = \mathcal{I} \dot{\cup} \mathcal{H} \dot{\cup} \mathcal{O}$ represents the set of all neurons being the disjunctive union of sets $\mathcal{I}$, $\mathcal{H}$ and $\mathcal{O}$. Be $\mathcal{I}$ the set of input neurons whose activations are externally given, $\mathcal{H}$ the set of hidden neurons whose activations are computed within the net and $\mathcal{O}$ the set of output neurons whose computed activations represent the output of the net.
2. Each of the n neurons N_i is characterized by an index i ($i \in \{1, \ldots, n\}$), its threshold θ_i and its activation a_i.
3. The network structure is given by the set $\mathcal{W}$ of weighted feed-forward links w_{ij} which interconnect two neurons N_i and N_j of adjacent layers. The set of links leading towards a neuron N_i is denoted $\mathcal{W}_i$.
4. In the case of hidden or output neurons, the activation a_i of a neuron N_i is calculated from the activation of preceding neurons as follows.

$$\forall i \in \{1, \ldots, n\} : N_i \in \mathcal{H} \cup \mathcal{O} \;\rightarrow\; a_i \;=\; \frac{1}{1 + e^{-(net_i - \theta_i)}}$$

$$net_i \;=\; \sum_{k \mid w_{ki} \in \mathcal{W}_i} w_{ki} \cdot a_k \;.$$

The activation of input neurons will be given externally and ranges from 0 to 1.

■

2.3 Neural conjunction

If all weights of the links leading to a neuron are below the neuron's threshold but the sum of these weights exceeds the threshold then this neuron will only fire when all connected neurons of the preceding layer are activated to 1. Thus, this neuron perfoms a logical conjunction of the connected inputs. Since we don't consider binary operations any more, we demand the activation to be at least 0.9 for a true conjunction and less than 0.1 for a false conjunction.

$$\frac{1}{1+e^{-(net_i-\theta_i)}} \geq 0.9$$

This demand can be transformed and net_i can be substituted by the equation from definition 2.2 part 4 which leads to the following equation.

$$\theta_i - ln\left(\frac{1}{0.9}-1\right) \leq net_i = \sum_{w_{ji}\in\mathcal{W}_i} w_{ji}\cdot a_j$$

Since the activation a_j of all connected neurons is assumed to be 1 for the true conjunction we can conclude a constraint for the relation of weights w_{ij} and the neuron's threshold θ.

$$\sum_{w_{ji}\in\mathcal{W}_i} w_{ji} \geq \theta_i - ln(0.\overline{1})$$

Additionally, for the false conjunction we demand the neuron's activation to be below 0.1 as soon as at least one of the connected neuron's activations is 0.

$$\frac{1}{1+e^{-(net_i-\theta_i)}} \leq 0.1$$

By analog transformations as above the following equation results.

$$\sum_{w_{ji}\in\mathcal{W}'\subset\mathcal{W}_i} w_{ji} \leq \theta_i - ln(9)$$

Here, $\mathcal{W}'$ is a *proper* subset of $\mathcal{W}_i$ denoting all combinations of weights which lead to the conjunctive neuron except the complete combination which is supposed to result in an activation greater than 0.9. It deosn't matter whether the weights have different values as long as these criteria are met. Thus, we derive a constraint for conjunctive neurons ($ln(9) = -ln(0.\overline{1}) \simeq 2.19$).

Definition 2.3 A neuron N_i *belongs to the set of conjunctive neurons* $\mathcal{C}$ if and only if

$$\begin{aligned}\forall N_i \in \mathcal{N} \ (\ N_i \in \mathcal{C} \longleftrightarrow \ & \forall w_{ji} \in \mathcal{W}_i \ . \ \sum_{w_{ji}\in\mathcal{W}} w_{ji} > \theta_i + ln(9) \ \wedge \\ & \forall \mathcal{W}' \subset \mathcal{W} \ . \ \sum_{w_{ji}\in\mathcal{W}'} w_{ji} < \theta_i - ln(9) \ )\end{aligned}$$

■

This conjunction is no longer binary but real–valued, since we allow analog values in the interval $[0, 1]$ at the neuron's inputs. Thus, also the activation of the conjunctive neuron, being the output of the conjunction, ranges in the same interval. Table 1 shows the real–valued activation of a binary and a neural conjunction for binary input values.

As one can see, the conjunction works well for binary inputs and for analog input values we get a smooth characteristic rather than a crisp one that is depicted in Figure 2. Figure 2 shows the similarity of the neural conjunction (c) with the two operators minimum (a) and algebraic product (b) which are commonly used in fuzzy logic.

input 1	input 2	binary conjunction	neural conjunction
0	0	0	0.00
0	1	0	0.01
1	0	0	0.01
1	1	1	0.99

Table 1: Neural activation a_i for input weights $w_{1i} = w_{2i} = 10$ and threshold $\theta_i = 15$ to represent the neural conjunction.

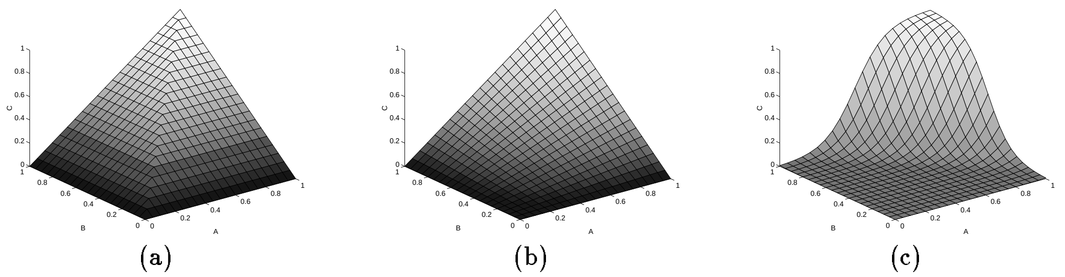

Figure 2: Representation of the conclusion C of an implication $A \wedge B \rightarrow C$ in fuzzy logic by the minimum operator (a) resp. the algebraic product (b) and in neural representation (c).

2.4 Neural disjunction

In a similar way we can have a neuron perform a disjunction. If every weight leading to a neuron exceeds its threshold the neuron will be activated as soon as one of the connected neurons has an activation of 1, thus performing a disjunction of its inputs. In analogy to the conjunction we demand the disjunctive neuron's activation to be above 0.9 as soon as one of the connected neurons has activation 1.

$$\forall w_{ji} \in \mathcal{W}_i \ . \ w_{ji} \geq \theta_i + ln(9)$$

Otherwise, if all connected neurons have activation 0, we demand an activation below 0.1 for the disjunctive neuron. This directly leads to a constraint for the neuron's threshold irrespectable of the weights.

$$\theta_i \geq ln(9)$$

From this we derive the following definition.

Definition 2.4 A neuron N_i *belongs to the set* $\mathcal{D}$ *of disjunctive neurons* if and only if

$$\forall N_i \in \mathcal{N} \ \ (\ N_i \in \mathcal{D} \ \longleftrightarrow \ \theta_i \geq ln(9) \ \wedge \ \forall w_{ji} \in \mathcal{W}_i \ . \ w_{ji} \geq \theta_i + ln(9))$$

■

Table 2 represents the real–valued activations of a disjunctive neuron for binary input values. Just as in the case of the conjunction, Figure 3 shows the similarity between a neural disjunction (c) and the common fuzzy operators maximum (a) and algebraic sum (b).

2.5 DNF– and CNF–networks

Since any quantifier–free logical formula can be represented in conjunctive reps. disjunctive normal form (see [8], p.19ff), we introduce a network architecture that is structured like such a formula in normal form. We will call this general form a *logical normal form network* (LNF–network) to summarize *conjunctive* and *disjunctive normal form networks* (CNF– and DNF–networks).

input 1	input 2	binary disjunction	neural disjunction
0	0	0	0.00
0	1	1	0.99
1	0	1	0.99
1	1	1	0.99

Table 2: Neural activation a_i for weights $w_{1i} = w_{2i} = 20$ and threshold $\theta_i = 15$ to represent a neural disjunction.

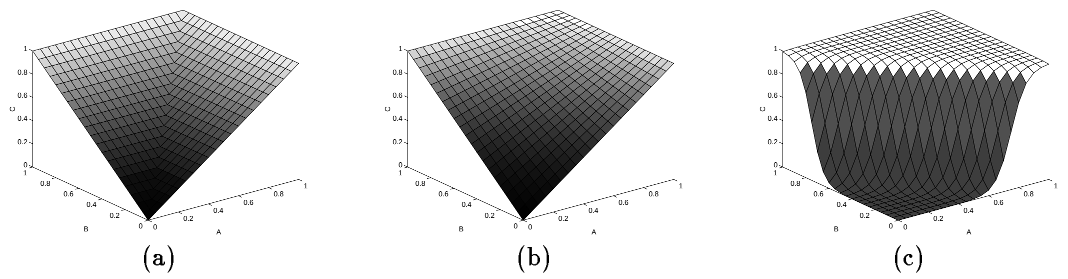

Figure 3: Representation of the conclusion C of the implication $A \vee B \rightarrow C$ in fuzzy logic by the maximum operator (a) resp. the algebraic sum (b) and in neural representation (c).

By applying the two deMorgan's thoerems that also hold for all pairs of t–norms and t–conorms in fuzzy logic (see [5], p. 25)

$$\begin{aligned} \neg(a \wedge b) &= \neg a \vee \neg b \\ \neg(a \vee b) &= \neg a \wedge \neg b \end{aligned} \tag{1}$$

we can transform the negation operator from outside a conjunction or disjunction to the inside, thus negating an atom. Therefore, we can create formulae which only contain literals, conjunctions and disjunctions. Negations will only occur within a literal. As is shown in [4] the negation of fuzzy variables is explicitely contained within the fuzzy representation if you obey the $\sum 1$ SEN design criterion[1]. These formulae can be represented in CNF– resp. DNF–networks that have such fuzzy membership functions as inputs without the need for negation inside the network architecture.

Definition 2.5 A neural network with three layers is called a *CNF–network* if and only if the hidden layer solely consists of disjunctive neurons and the output layer consists of exactly one conjunctive neuron. ■

Definition 2.6 A neural network with three layers is called a *DNF–network* if and only if the hidden layer solely consists of conjunctive neurons and the output layer consists of exactly one disjunctive neuron. ■

Details of CNF– and DNF–networks especially regarding additional constraints and proofs of theorems can be found in [4].

2.6 The learning algorithm

Since the standard backpropagation algorithm would violate the constraints for conjunctive resp. disjunctive neurons with its new weight calculations, we had to adapt the learning algoritm. In "vanilla backpropagation" weight changes Δw_{ij} are calculated as follows [7]

$$\Delta w_{ij} = \eta \cdot o_i \cdot \delta_j \tag{2}$$

[1]According to this criterion, fuzzy membership functions have to be continuous, have to sum up to 1 in any point of the universe and have to obey the an extended version of the normalization criterion.

where the output o_i of neuron N_i is identical to its activation a_i , η represents the learning rate and

$$\delta_i = \begin{cases} f_j'(net_j) \cdot (t_j - a_i) & \text{if } N_j \in \mathcal{O} \\ f_j'(net_j) \cdot \sum_k \delta_k w_{jk} & \text{if } N_j \in \mathcal{H} \ . \end{cases}$$

Here, $f_j'(net_j)$ is the derivation of the sigmoidal activation function (see definition 2.2 part 4) for neuron N_j , and t_j is the target value of the output neuron that is supposed to be generated by the net.

In contrast to the ordinary initialization function that randomly asserts values to thresholds and weights, we created a special initialization function for our network architecture. This algorithm initializes weights and threshold so they obey the criteria in definitions 2.3 and 2.4. Since these criteria still leave some degrees of freedom, the weight values are chosen randomly within an ϵ–interval $[x - \epsilon, x + \epsilon]$ around a randomly chosen point x that obeys the criteria. All weights are assigned positive values.

The algorithm to train logical normal form networks, LNF–backpropagation, calculates the weight changes according to definition 2.7.

Definition 2.7 *LNF–Backpropagation* is performed as follows:

1. First of all, propositions for the weight changes are calculated using standard backpropagation according to equation (2).

2. In the case of hidden neurons the neuron is detected that best resembles the desired output according to the winner–take–all principle (see [5], p. 33) For this neuron the learning rate η remains as selected — for all other neurons of this layer η will be multiplied by the ratio of preselected learning rate to number of neurons in this layer. Thus, we call this the *winner–take–most* principle. The learning rate for output neurons remains unchanged.

3. In the next step the proposed weight changes are performed while two additional constraints must be satisfied:

   a) If the new weight is too small to obey the criteria for logical neurons (definition 2.3 or 2.4) it will be clipped to 0. These weight will no longer contribute to the network output.

   b) If the new weight is too large to to obey the criteria for logical neurons (definition 2.3 or 2.4) it will be clipped to the maximum allowed value.

■

If a fuzzy–logically interpretable LNF–network is trained with this learning algorithm and if the error is close to 0 after training we can extract a fuzzy–logical formula from the structure of the net and obtain the same result as in the net with a rule–based approach using these formulae. Hence, we gain the explanation ability which is demanded in many applications.

hidden	input N_1	N_2	N_3	N_4	conjunctions of hidden layer	output N_7
N_5	10	–	–	10	$N_1 \wedge N_4$	15
N_6	–	10	10	–	$N_2 \wedge N_3$	15
disjunction of output layer:						$N_5 \vee N_6$

Table 3: The weight matrix of a fuzzy–logically interpretable DNF–network that represents an XOR.

The weights shown in Table 3 represent an XOR of the fuzzy variables A and B as shown in Figure 1. Neuron N_1 represents A while N_2 represents $\neg A$ ($N_3 = B, N_4 = \neg B$). Neuron N_5 represents the conjunction of input neurons N_1 and N_4 ($A \wedge \neg B$) since it is no longer connected to the other input units. Weights w_{25} and w_{35} were eliminated and depicted with a dash. Neuron N_6 represents the conjunction $\neg A \wedge B$ due to its connections to input units N_2 and N_3 . Weights w_{16} and w_{46} were eliminated. Neuron N_7 finally represents the disjunction of the two conjunctive hidden neurons N_5 and N_6 . Thus, we can extract the DNF–formula $(A \wedge \neg B) \vee (\neg A \wedge B)$ which resembles the XOR in DNF–notation.

3 Conclusions

Amongst the variety of existing approaches to extract rule–based knowledge form unrestricted neural networks we presented an architecture that is restricted to perform logical operations. Although approaches like the M–of–N algorithm [9], RuleNet [6] and RULEX [2] claim to extract *propositional–type* IF THEN rules, their rules cannot directly be interpreted in logical normal form and be processed elsewhere. Our rules instead can directly be fed into a rule–based reasoning system (expert system) that then similarly performs the former task of the network. Additionally, we allow the inverse direction of knowledge transfer: a–priori–knowledge can be mapped into the net prior to training.

References

[1] R. Andrews, J. Diederich, and A.B. Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6), 1995.

[2] R. Andrews and S. Geva. Rule extraction from a constrained error back propagation mlp. In *Australian Conference on Neural Networks*, 1994.

[3] Y. Le Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. In R.P. Lippman, J.E. Moody, and D. S. Touretzky, editors, *Proceedings of the 2nd Conference on Advances in Neural Information Processing Systems (NIPS-90)*, pages 598–605, 1990.

[4] C.S. Herrmann. *A hybrid AI–system for medical diagnosis: using the electroencephalogram as an example.* PhD thesis, TH Darmstadt, FG Intellektik, 1996.

[5] R. Kruse, J. Gebhardt, and F. Klawonn. *Fuzzy-Systeme.* Teubner, 1995.

[6] C. McMillan, M. Mozer, and P. Smolensky. The connectionist scientist game: rule extraction and refinement in a neural network. In *Thirteenth Annual Conference of the Cognitive Science Society*, 1991.

[7] D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT Press, 1986.

[8] U. Schöning. *Logic for Computer Scientists.* Birkhäuser, 1989.

[9] G. Towell and J. Shavlik. The extraction of refined rules from knowledge based neural networks. In *Machine Learning*, volume 131, pages 71–101, 1993.

[10] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.