

# Neural Networks and Rational Łukasiewicz Logic

Paolo Amato   Antonio Di Nola   Brunella Gerla

*Abstract*—In this paper we shall describe a correspondence between Rational Łukasiewicz formulas and neural networks in which the activation function is the truncated identity and synaptic weights are rational numbers. On one hand to have a logical representation (in a given logic) of neural networks could widen the interpretability, amalgamability and reuse of these objects. On the other hand, neural networks could be used to learn formulas from data and as circuitual counterparts of (functions represented by) formulas.

## I. INTRODUCTION

Neural networks are one of the main constituents of Soft Computing methodologies. Their ability to learn from data has made them valuable tools in such diverse applications as modeling, time series analysis, pattern recognition, signal processing, and control. In particular, neural networks have a great deal to offer when the solution of the problem of interest is made difficult by the lacking or untractability of a mathematical description, or the data are generated by a complex (highly nonlinear) process. For these reasons, neural networks became a multidisciplinary subject with roots in neurosciences, mathematics, statistics, physics, computer science, and engineering. Unfortunately until now logic seems to not belong to these roots.

The lack of a deep investigation of the relationships between logic and neural networks is somewhat astonishing. In fact both the fields could gain advantages from this investigation. On one hand to have a logical representation (in a given logic) of neural networks could widen the interpretability, amalgamability and reuse of these objects. On the other hand, neural networks could be used to learn formulas from data and as circuitual counterparts of (functions represented by) formulas. In fact they are either easy to implement and high parallel objects.

In [2] it is shown that, by taking as activation function  $\psi$  the identity truncated to zero and one (i.e.,  $\psi(x) = (1 \wedge (x \vee 0))$ ), it is possible to represent the corresponding neural network as combination of propositions of Łukasiewicz calculus (and *viceversa*). But what is missing in that work is the identification of the logic to which these linear combinations belong, and the analysis of the consequences of this equivalence.

In this work we show that neural networks whose activation function is the identity truncated to zero and one, are equivalent to (equivalence classes of) formulas of Rational Łukasiewicz logic.

P. Amato: ST Microelectronics, Via C. Olivetti 2, 20041 Agrate Br. (MI), Italy, e-mail: [paolo.amato@st.com](mailto:paolo.amato@st.com)

A. Di Nola: Dept. Mathematics and Informatics, University of Salerno, Via S. Allende, 84081 Baronissi (SA), Italy, e-mail: [dinola@unisa.it](mailto:dinola@unisa.it)

B. Gerla: Dept. Mathematics and Informatics, University of Salerno, Via S. Allende, 84081 Baronissi (SA), Italy, e-mail: [bgerla@unisa.it](mailto:bgerla@unisa.it)

## II. A BRIEF DESCRIPTION OF RATIONAL ŁUKASIEWICZ LOGIC

Rational Łukasiewicz logic has been introduced in [4] as an extension of Łukasiewicz logic.

Formulas of Rational Łukasiewicz propositional calculus are built from the connectives of disjunction ( $\oplus$ ), negation ( $\neg$ ) and division ( $\delta_n$ ) in the usual way.

Interpretation of connectives of Rational Łukasiewicz logic is given by the following definition.

*Definition II.1:* An assignment is a function  $v : Form \rightarrow [0, 1]$  such that

- $v(\neg\varphi) = 1 - v(\varphi)$
- $v(\varphi \oplus \psi) = \min(1, v(\varphi) + v(\psi))$ .
- $v(\delta_n\varphi) = \frac{v(\varphi)}{n}$ .

Every function  $\iota$  from the set of variables to  $[0, 1]$  is uniquely extendible to an assignment  $v^\iota$ . For each point  $\mathbf{x} = (x_1, \dots, x_n) \in [0, 1]^n$  let  $\iota_{\mathbf{x}}$  be the function mapping each variable  $X_j$  into  $x_j$ . Fixed  $n$ , with each formula  $\varphi$  with  $|\text{var}(\varphi)| \leq n$  it is possible to associate the function

$$f_\varphi : \mathbf{x} \in [0, 1]^n \mapsto v^{\iota_{\mathbf{x}}}(\varphi) \in [0, 1]$$

satisfying the following conditions:

- $f_{X_i}(x_1, \dots, x_n) = x_i =$  the  $i$ th projection.
- $f_{\neg\varphi} = 1 - f_\varphi$ .
- $f_{(\varphi \oplus \psi)} = \min(1, f_\varphi + f_\psi)$
- $f_{(\delta_n\varphi)} = \frac{f_\varphi}{n}$ .

The function  $f_\varphi$  is the *truth table* of the formula  $\varphi$ . Formula  $n\varphi$  is the multiple of  $\varphi$  with respect to  $\oplus$ , defined recursively by  $2\varphi = \varphi \oplus \varphi$  and  $n\varphi = (n-1)\varphi \oplus \varphi$ .

The following connectives will be useful in the sequel:

$$\begin{aligned} x \odot y &= \neg(\neg x \oplus \neg y), \\ x \vee y &= (x \odot \neg y) \oplus y, \\ x \wedge y &= \neg(\neg x \vee \neg y). \end{aligned}$$

They are interpreted by the following truth tables:

$$\begin{aligned} f_{\alpha_1 \odot \alpha_2} &= \max(f_{\alpha_1} + f_{\alpha_2} - 1, 0), \\ f_{\alpha_1 \vee \alpha_2} &= \max(f_{\alpha_1}, f_{\alpha_2}), \\ f_{\alpha_1 \wedge \alpha_2} &= \min(f_{\alpha_1}, f_{\alpha_2}). \end{aligned}$$

A formula  $\varphi$  with  $|\text{var}(\varphi)| < n$  is *satisfiable* iff there exists  $\mathbf{x} \in [0, 1]^n$  such that  $f_\varphi(\mathbf{x}) = 1$ .  $\varphi$  is a *tautology* iff for every  $\mathbf{x} \in [0, 1]^n$ ,  $f_\varphi(\mathbf{x}) = 1$ . An assignment  $v$  is a *model* of a set of formulas  $\Gamma$  if for every  $\tau \in \Gamma$ ,  $v(\tau) = 1$ .

Characterization of truth tables of many-valued formulas is a key tool for a deep understanding of the logics and for the use of formulas as approximators (see [1],[3]).

*Definition II.2:* A Rational McNaughton function is a continuous piecewise linear function such that each piece has rational coefficients.

A direct inspection shows that every function  $f_\varphi$  is a Rational McNaughton function.

In [6] (see [7] for a constructive proof) it has been shown that a function is associated with a Lukasiewicz formula if and only if it is a continuous piecewise linear function, each piece having integer coefficients. Analogously we can prove that truth tables of Rational Lukasiewicz formulas are the totality of Rational McNaughton functions.

At least two different ways of representation exist to prove that every piecewise linear function  $f : [0, 1]^n \rightarrow [0, 1]$  with rational coefficients is the truth table of some Rational Lukasiewicz formula:

- The first way is to decompose the function  $f$  in some basic pieces  $h_v$  (Schauder hats) that can be associated with some formula, in such a way that

$$f(x_1, \dots, x_n) = \sum_{v \in V} h_v(x_1, \dots, x_n). \quad (1)$$

- The second approach is based on proving that each hyperplane with rational coefficients is the truth table of a formula, and then showing that there exist hyperplanes  $p_{i,j}$  such that

$$f(x_1, \dots, x_n) = \bigvee_{i \in I} \bigwedge_{j \in J} p_{i,j}(x_1, \dots, x_n). \quad (2)$$

While the first method (1) is more related with the expression of neural nets (see Equation (3)), it comes out to be complicate even in the case of two variables: indeed founding formulas associated with Schauder hats  $h_v$  can be as hard as the same problem for the whole function  $f$ .

In next section we shall give some details on the case of one variable using the first approach, and then we shall treat the case of more variables using the second approach.

### III. (RATIONAL) NEURAL NETWORKS ARE EQUIVALENT TO FORMULAS OF RATIONAL LUKASIEWICZ LOGIC

Among the many possible neural networks typologies and structures, we focus our attention on multilayer perceptrons. These are feedforward neural networks with one or more hidden layers. A multilayer perceptron with  $l$  hidden layers,  $n$  inputs and one output can be represented as a function  $F : [0, 1]^n \rightarrow [0, 1]$  [5] such that  $F(x_1, \dots, x_n) =$

$$\phi \left( \sum_{k=1}^{n^{(l)}} \omega_{ok} \phi \left( \sum_{j=1}^{n^{(l-1)}} \omega_{kj} \phi \left( \dots \left( \sum_{i=1}^n \omega_{li} x_i + b_i \right) \dots \right) \right) \right),$$

where  $\phi : \mathbb{R} \rightarrow [0, 1]$  is a monotone-nondecreasing continuous function (referred to as activation function),  $\omega_{ok}$  is the synaptic weight from neuron  $k$  in the  $l$ -th hidden layer to the single output neuron  $o$ ,  $\omega_{kj}$  is the synaptic weight from neuron  $j$  in the  $(l-1)$ -th hidden layer to neuron  $k$  in the  $l$ -th hidden layer, and so on for the other synaptic weights. A graphical representation of a multilayer perceptron is given by Figure 1.

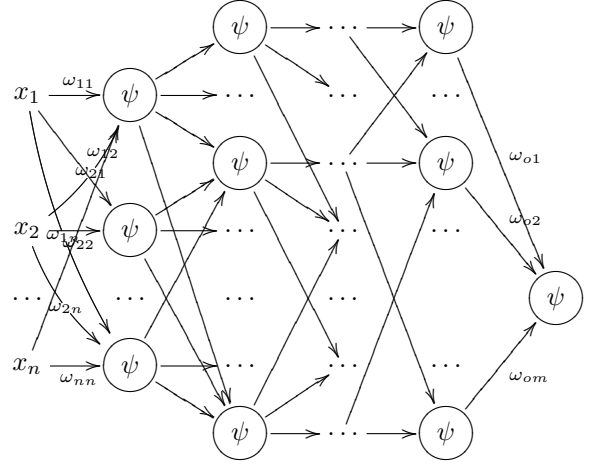


Fig. 1. Graphical representation of a multilayer perceptron.

In the simplest case, a multilayer perceptron has exactly one hidden layer. This network can be represented as a function  $G : [0, 1]^n \rightarrow [0, 1]$ :

$$G(x_1, \dots, x_n) = \sum_{i=1}^{\bar{n}} \alpha_i \phi \left( \sum_{j=1}^n w_{ij} x_j + b_i \right), \quad (3)$$

where  $\bar{n}$  is the number of neurons in the hidden layer.

In the following we shall consider multilayer perceptrons where the activation function is the piecewise linear function  $\psi(x) = \max(\min(1, x), 0)$ , and the synaptic weights are rational numbers.

#### A. The one dimensional case

Given a subset  $\{(r_1, s_1), \dots, (r_n, s_n)\}$  of  $(\mathbb{Q} \cap [0, 1])^2$ , the family of *rational (Schauder) hats* associated with  $\{(r_1, s_1), \dots, (r_n, s_n)\}$  is a family of continuous piecewise linear functions  $h_{r_i}^{s_i} : [0, 1] \rightarrow [0, 1]$ , such that

$$h_{r_i}^{s_i}(r_j) = \begin{cases} s_i & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases}$$

and  $h_{r_i}^{s_i}$  is linear in every interval  $[r_j, r_{j+1}]$  for  $j = 1, \dots, n-1$ .

Let  $f : [0, 1] \rightarrow [0, 1]$  be a continuous, piecewise linear function with rational coefficients. Let  $\{(r_1, f(r_1)), \dots, (r_u, f(r_u))\}$  be the set of points in which  $f$  is not differentiable. Then

$$f(x) = \sum_{i=1}^u h_{r_i}^{f(r_i)}(x).$$

*Lemma III.1:* Let  $h_{r_i}^{s_i}$  (with  $1 < i < n$ ) be a rational hat associated with the set  $\{(r_1, s_1), \dots, (r_n, s_n)\}$ . Let  $\psi : \mathbb{R} \rightarrow [0, 1]$  be defined as  $\psi(x) = (1 \wedge x) \vee 0$ . Then  $H : [0, 1] \rightarrow [0, 1]$  defined as:

$$H(x) = s_i \psi(1) - s_i \psi \left( \frac{-1}{r_i - r_{i-1}} x + \frac{r_i}{r_i - r_{i-1}} \right)$$

$$- s_i \psi \left( \frac{1}{r_{i+1} - r_i} x - \frac{r_i}{r_{i+1} - r_i} \right),$$

is such that  $h_{r_i}^{s_i} = H$ .

*Proof:* The hat  $h_{r_i}^{s_i}$  has the form:

$$h_{r_i}^{s_i}(x) = \begin{cases} 0 & \text{for } x \in [0, r_{i-1}] \cup [r_{i+1}, 1] \\ s_i \frac{x - r_{i-1}}{r_i - r_{i-1}} & \text{for } x \in [r_{i-1}, r_i] \\ s_i \frac{r_{i+1} - x}{r_{i+1} - r_i} & \text{for } x \in [r_i, r_{i+1}] \end{cases}$$

Let  $\Delta_l(x) = \frac{-1}{r_i - r_{i-1}}x + \frac{r_i}{r_i - r_{i-1}}$ , and  $\Delta_r(x) = \frac{1}{r_{i+1} - r_i}x - \frac{r_i}{r_{i+1} - r_i}$ . Then  $H(x) = s_i - s_i \psi(\Delta_l(x)) - s_i \psi(\Delta_r(x))$ . Now there are four cases:

- When  $x \in [0, r_{i-1}]$ ,  $\Delta_l(x) \geq 1$  and  $\Delta_r(x) < 0$ . Then  $H(x) = 0$ .
- When  $x \in [r_{i-1}, r_i]$ ,  $0 \leq \Delta_l(x) \leq 1$  and  $\Delta_r(x) < 0$ . Then  $H(x) = s_i - s_i \frac{r_i - x}{r_i - r_{i-1}} = s_i \frac{x - r_{i-1}}{r_i - r_{i-1}}$ .
- When  $x \in [r_i, r_{i+1}]$ ,  $\Delta_l(x) < 0$  and  $0 \leq \Delta_r(x) \leq 1$ . Then  $H(x) = s_i \frac{r_{i+1} - x}{r_{i+1} - r_i}$ .
- When  $x \in [r_{i+1}, 1]$ ,  $\Delta_l(x) < 0$  and  $\Delta_r(x) \geq 1$ . Then  $H(x) = 0$ . ■

The above Lemma can be easily extended to Schauder hats  $h_{r_1}^{s_1}$  and  $h_{r_n}^{s_n}$ .

In the following we shall refer to nets like  $H$  as (*Schauder*) *hat nets*, because they shall play for neural networks the same role that rational hats play in Rational Lukasiewicz logic.

*Theorem III.2:* Let  $\psi : \mathbb{R} \mapsto [0, 1]$  be defined as  $\psi(x) = (1 \wedge x) \vee 0$ . Then:

- (i) For all  $\bar{n} \in \mathbb{N}$ ,  $\alpha_i, w_{ij}, b_i \in \mathbb{Q}$ , where  $i = 1, \dots, \bar{n}$ ,  $F : [0, 1] \mapsto [0, 1]$  defined as:

$$F(x) = \sum_{i=1}^{\bar{n}} \alpha_i \psi(w_i x + b_i)$$

is a rational McNaughton function.

- (ii) For any rational McNaughton function  $f$ , there exist  $\bar{n} \in \mathbb{N}$  and  $\alpha_i, w_i, b_i \in \mathbb{Q}$ , where  $i = 1, \dots, \bar{n}$  and  $j = 1, \dots, n$ , such that

$$f(x_1, \dots, x_n) = \sum_{i=1}^{\bar{n}} \alpha_i \psi(w_i x + b_i).$$

*Proof:* (i). The function  $\psi$  is a piecewise linear function and  $\psi(w_i x + b_i)$  is a piecewise linear function. Since a sum of a finite number of piecewise linear function is a piecewise linear function,  $\sum_{i=1}^{\bar{n}} \alpha_i \psi(w_i x + b_i)$  is again a piecewise linear function taking values in  $[0, 1]$ . Hence it is a rational McNaughton function.

(ii). Every rational McNaughton function  $f$  can be represented as a weighted sum of rational hats. Since from Lemma III.1 it is possible to associate to each hat  $h$  a hat net  $H$ ,  $f$  can be represented by a weighted hat nets. ■

## B. The $n$ -dimensional case

In order to extend Theorem III.2 to the  $n$  dimensional case, we need some preliminary results:

*Lemma III.3:* The activation function  $\psi$  maps any finite weighted sum of rational McNaughton functions (where the weight are rational values) into a rational McNaughton function.

*Proof:* Let  $f, f_1, f_2$  be rational McNaughton functions and  $p, p_1, p_2, q, q_1, q_2$  be natural numbers. Then it is easy to check that:

$$\begin{aligned} \psi(f) &= f \\ \psi(f_1 + f_2) &= f_1 \oplus f_2 \\ \psi(f_1 - f_2) &= f_1 \ominus f_2 \\ \psi\left(\frac{p}{q}f\right) &= p\delta_q f \\ \psi\left(\frac{p_1}{q_1}f_1 - \frac{p_2}{q_2}f_2\right) &= p_1\delta_{q_1}f_1 \ominus p_2\delta_{q_2}f_2 \end{aligned}$$

We want now to associate a neural network to each Rational Lukasiewicz formula of  $n$  variables. We could extend the concept of Schauder hats to the  $n$ -dimensional case, but as already mentioned, the description for such hats is very complicate. We shall give an alternative method based on the proof of McNaughton representation theorem as presented in [1].

Let  $f : [0, 1]^n \rightarrow [0, 1]$  be a piecewise linear function and let  $p_1, \dots, p_u$  be the linear functions that are the *pieces* of  $f$  in the sense that for every  $\mathbf{x} \in [0, 1]^n$  there exists  $i \in \{1, \dots, u\}$  such that  $f(\mathbf{x}) = p_i(\mathbf{x})$ .

Let  $\Sigma$  denote the set of permutations of  $\{1, \dots, u\}$  and for every  $\sigma \in \Sigma$  let

$$P_\sigma = \{\mathbf{x} \in [0, 1]^n \mid p_{\sigma(1)}(\mathbf{x}) \leq \dots \leq p_{\sigma(u)}(\mathbf{x})\}.$$

In other words  $P_\sigma$  is a polyhedron such that the restrictions of linear functions  $p_1, \dots, p_u$  to  $P_\sigma$  are totally ordered, increasingly with respect to  $\{\sigma(1), \dots, \sigma(u)\}$ . We denote by  $i_\sigma$  the index such that

$$f(\mathbf{x}) = p_{\sigma(i_\sigma)}(\mathbf{x}) \quad \text{for every } \mathbf{x} \in P_\sigma.$$

The proof of the following theorem can be found in [1].

*Theorem III.4:* For every  $\mathbf{x} \in [0, 1]^n$ ,

$$f(\mathbf{x}) = \min_{\sigma \in \Sigma} \max_{1 \leq j \leq i_\sigma} p_{\sigma(j)}(\mathbf{x}) = \min_{\sigma \in \Sigma} \max_{1 \leq j \leq i_\sigma} \psi(p_{\sigma(j)}(\mathbf{x})).$$

By using neural networks we can express linear combinations, but we need to define networks corresponding to minimum and maximum.

*Proposition III.5:* For every  $x, y \in [0, 1]$ , one-layer neural networks

$$\begin{aligned} F_1(x, y) &= \psi(y) - \psi(y - x) \\ F_2(x, y) &= \psi(y) + \psi(x - y) \end{aligned}$$

coincides respectively with  $\min(x, y)$  and  $\max(x, y)$ .

*Proof:* If  $x \leq y$  then  $\psi(y) = y$ ,  $\psi(y - x) = y - x$  and  $\psi(x - y) = 0$ , hence  $F_1(x, y) = x$  and  $F_2(x, y) = y$ . ■

If  $y \leq x$  then  $\psi(y) = y$ ,  $\psi(y-x) = 0$  and  $\psi(x-y) = x-y$ , hence  $F_1(x, y) = y$  and  $F_2(x, y) = x$ . ■

We can hence describe neural representation of McNaughton functions.

*Theorem III.6:* (i) For every  $l, n, n^{(2)}, \dots, n^{(l)} \in \mathbb{N}$ , and  $\omega_{ij}, b_i \in \mathbb{Q}$ , the function  $F : [0, 1]^n \mapsto [0, 1]$  defined as  $F(x_1, \dots, x_n) =$

$$\phi \left( \sum_{k=1}^{n^{(l)}} \omega_{ok} \phi \left( \sum_{j=1}^{n^{(l-1)}} \omega_{kj} \phi \left( \dots \left( \sum_{i=1}^n \omega_{li} x_i + b_i \right) \dots \right) \right) \right),$$

is a rational McNaughton function.

(ii) For any rational McNaughton function  $f$ , there exist  $l, n, n^{(2)}, \dots, n^{(l)} \in \mathbb{N}$ , and  $\omega_{ij}, b_i \in \mathbb{Q}$  such that  $f(x_1, \dots, x_n) =$

$$\phi \left( \sum_{k=1}^{n^{(l)}} \omega_{ok} \phi \left( \sum_{j=1}^{n^{(l-1)}} \omega_{kj} \phi \left( \dots \left( \sum_{i=1}^n \omega_{li} x_i + b_i \right) \dots \right) \right) \right).$$

*Proof:* (i) By Lemma III.3.

(ii) By Theorem III.4 we have

$$f(\mathbf{x}) = \min_{\sigma \in \Sigma} \max_{1 \leq j \leq i_\sigma} \psi(p_{\sigma(j)}(\mathbf{x})).$$

For every  $\sigma$  and  $j$ , the function  $\psi(p_{\sigma(j)}(\mathbf{x}))$  is a network with one hidden layer. Then applying networks as in Proposition III.5 we get the claim. ■

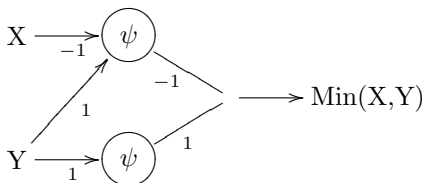
*Corollary III.7:* The class of neural networks as in Theorem III.6 is dense in the class of multi-layer perceptrons representing continuous functions.

*Proof:* By using a simple variation of Weierstrass theorem [8] it is possible to show that rational McNaughton functions are able to approximate every continuous function with an error as low as desired. Then, by the previous theorem we get the claim. ■

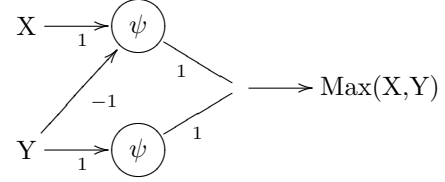
From the corollary it follows that the use of only truncated identity  $\psi$  as activation function is not a severe restriction on the the class of neural networks which can be obtained; they can approximate every neural network representing a continuous function.

### C. Examples

In this section we shall give some examples of the previous theorem. First of all note that networks corresponding to  $\min(x, y)$  and  $\max(x, y)$  are given by



and



1. Consider the formula  $(x \oplus y) \odot (\neg(x \oplus y) \oplus 2x)$ , whose truth table is given by Figure 2.

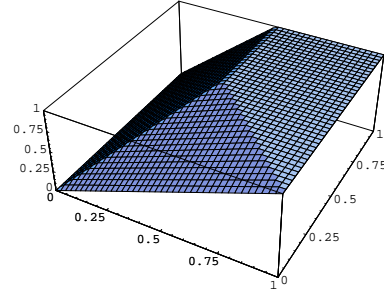


Fig. 2. The rational McNaughton function associated to the formula  $(x \oplus y) \odot (\neg(x \oplus y) \oplus 2x)$

Such function is equal to  $(3x \wedge 1) \wedge ((x + y) \wedge 1)$  hence the corresponding neural network will be as in Figure 3.

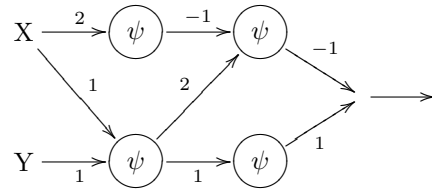


Fig. 3. The neural network representing the formula  $(x \oplus y) \odot (\neg(x \oplus y) \oplus 2x)$

### 2. The McNaughton function

$$((x \oplus \delta_3 x) \vee 2y) \wedge (2x \vee 2\neg y)$$

whose graphic is given by Figure 4, corresponds to neural network in Figure 5.

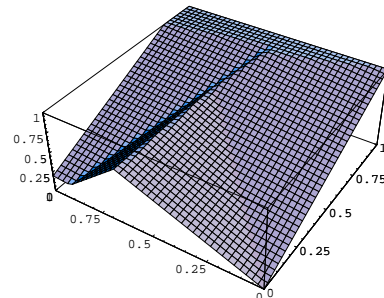


Fig. 4. Graphic of  $((x \oplus \delta_3 x) \vee 2y) \wedge (2x \vee 2\neg y)$

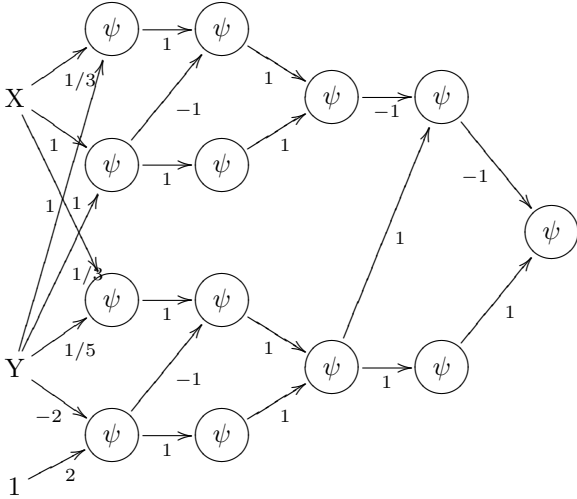


Fig. 5. Neural network corresponding to function of Figure 4

#### IV. CONCLUSIONS

The learning process is one of the most distinguished feature of NN. However there are some drawbacks. Learning has a cost. Thus when we deal with complex systems, we need to reduce this cost.

We can use a NN to learn a model. But we have no tools to use what has been obtained for simplifying the learning of a slightly different model. For example we could think that the synaptic weights of the NN associated to this new model are for the most part the same of the old NN. However we have nothing to do, but to start from scratch. Each learning has an history *per se*, we cannot learn from learning.

Moreover many times, beside a set of examples (like experimental data), other information (partial mathematical model, linguistic description) is available about the system we want to model. It should be very useful to use this kind of information to reduce the search space for the optimal network. Unfortunately actually there are no well defined methodologies suitable for incorporate *a priori* information in the network structure. At most there exist some *ad hoc* highly specific procedures

In this paper we shown that it is possible to represent NN with a saturating linear transfer function and rational weights as formulas of a given logic (rational Łukasiewicz logic) and viceversa. In this way we can integrate different kind of knowledge in the NN and divide a NN in subnetworks, each responsible of a certain behavior.

Moreover, from the point of view of logic, now it is possible to associate to each formula an analog circuit. Thus, although we did not yet define a normal form, we made a step in this direction and gave the possibility to physically implement the formulas of rational Łukasiewicz logic.

#### REFERENCES

[1] S. Aguzzoli. *Geometrical and Proof Theoretical Issues in Łukasiewicz propositional logics*. PhD thesis, University of Siena, Italy, 1998.

[2] J.L. Castro and E. Trillas. The logic of neural networks. *Mathware and Soft Computing*, 5:23–27, 1998.

[3] B. Gerla. *Functional representation of many-valued logics based on continuous t-norms*. PhD thesis, University of Milano, Italy, 2000.

[4] B. Gerla. Rational Łukasiewicz logic and Divisible MV-algebras. *Neural Networks World*, 11:159, 2001.

[5] S. Haykin. *Neural Networks – A Comprehensive Foundation*. Prentice-Hall, London, 1999.

[6] R. McNaughton. A theorem about infinite-valued sentential logic. *The Journal of Symbolic Logic*, 16(1):1–13, 1951.

[7] D. Mundici. A constructive proof of McNaughton’s theorem in infinite-valued logics. *Journal of Symbolic logic*, 59:596–602, 1994.

[8] P. Amato and M. Porto. An algorithm for the automatic generation of a logical formula representing a control law. *Neural Network World*, 10(5):777–786, 2000.