IMS ALGORITHM FOR LEARNING REPRESENTATIONS IN BOOLEAN NEURAL NETWORKS

Nripendra N. Biswas, T.V.M.K.Murthy, and M.Chandrasekhar

Department of Electrical Communication Engineering
Indian Institute of Science
Bangalore - 560 012, INDIA

ABSTRACT

A new algorithm for learning representations in Boolean neural networks, where the inputs and outputs are binary bits, is presented. The algorithm has become feasible because of a newly discovered theorem which states that any non-linearly separable Boolean function can be expressed as a convergent series of linearly separable functions connected by the logical OR (+) and the logical INHIBIT (-) operators. The formation of the series is carried out by many important properties exhibited by the implied minterm structure of a linearly separable function. The learning algorithm produces the representation much faster than the back propagation, and unlike the latter does not encounter the problem of local minima. It also successfully separates a linearly separable function and obtains the perceptron solution in the presence of a spoiler vector, a situation where back propagation is guaranteed to fail.

1. INTRODUCTION

The back propagation (BP) alogrithm developed by Rumelhart et al [1] is the most extensively used algorithm for learning representations by multilayer systems. Among many networks where back propagation has been successfully applied, there is a class of networks called Boolean neural networks where the input and output vectors are strings of binary bits, 0 and 1. The representation obtained for such networks by the back propagation algorithm becomes a logic circuit implemented by the familiar threshold gates [2,3] working as hidden and output units. Throughout our discussion, we have assumed that a gate is normally off, having an output 0, and gets turned on producing an output 1, when the weighted sum exceeds the threshold shown in the circle symbolizing the gate. In the terminology of logical design and switching theory of computer science and engineering, the output vector becomes the output Boolean function, and the input state vectors which are binary combinations of input variables are known as fundamental products or minterms. The output Boolean function can, therefore, be depicted in a truth table or may be expressed as a sum of minterms [3]. Boolean functions which can be realized by a single threshold gate are linearly separable (LS) functions [2,3]. However, quite often the desired output function is not LS. In such a case the output function can be expressed as a convergent series of LS functions,

This research is supported by a grant from the Council of Scientific and Industrial Research, New Delhi, INDIA.

given by a new theorem (The Linerization theorem) stated in this paper. Based on this theorem and the implied minterm structure [5] of LS functions, a new learning algorithm for Boolean neural networks has been developed.

Theorem 1 : (The Linearization Theorem) A non-linearly separable Boolean function F can always be expressed as a convergent series of linearly separable (LS) functions $F_1, F_2, \ldots F_n$ connected by the logical OR (+) and the logical INHIBIT (-) operators, such that,

$F_1 > F_2 > \ldots > F_n$, and

$F = F_1 - F_2 + \ldots + F_n$ when n is odd, and

$F = F_1 - F_2 + \ldots - F_n$ when n is even.

The proof of this theorem has been given in [4].

2. IMPLIED MINTERM STRUCTURE

We assume that the reader is familiar with some of the basic terms of switching theory, such as the on-minterms, off-terms, dc-minterms, linearly separable(LS) function. We also assume that the reader is also familiar with the definitions and working of a threshold *gate* and the associated terms, such as weight vector, weighted sum, upper and lower bounds of threshold* realization vector of an LS function.

Definition 1 : The tabular form of a given function is the on-minterms written in their binary form. It may also be considered as a matrix where each row represents a minterm whose output is desired to be 1, and each column is headed by a variable. For example, the tabular form of a 4- variable function is shown in table I(a).

Definition 2: A Boolean function is called positive and ordered, if in every column of the tabular form of the given function, the number of 1s is not less than the number of 0s, and the ratios of the number of 1s to that of 0s are such that if r_i and r_j are these ratios of the ith and the jth columns respectively, then $r_i \geq r_j$ when $j > i$. Here, the ith column is towards the right of the jth column, and columns are counted from right to left, that is, the rightmost column is the first column.

An arbitrary Boolean function can be made positive and ordered by suitable permutations and/or complementations of the columns of its tabular form.

Definition 3 : A linearly separable function which is positive and ordered will be called a positive ordered LS (POLS) function.

In a POLS function, it has been shown [2,3,5] that if a minterm m_i is realized by a set of weights and threshold, then there are minterms which will also be realized by the same set. If m_j is one such minterm, then m_i is said to imply m_j. In a POLS function the implied minterm m_j is obtained from the implying minterm m_i by either unit postitive permutation or by elementary positive complementaion. These terms have been explained in [4,5]. Also, note that the implication relation is transitive. For this reason, all the minterms of a POLS function form a graphical structure where the implication relations between them are shown. Such a structure is called an implied minterm structure [IMS]. A detailed and brief discussion about the IMS can be found in [5] and [4] respectively. A modular structure very

(a)					(b)				
m_i	x_4	x_3	x_2	x_1	m_i	x_1	x_3	x_4	x'_2
3	0	0	1	1	8	1	0	0	0
4	0	1	0	0	5	0	1	0	1
5	0	1	0	1	13	1	1	0	1
7	0	1	1	1	12	1	1	0	0
9	1	0	0	1	11	1	0	1	1
13	1	1	0	1	15	1	1	1	1
14	1	1	1	0	6	0	1	1	0
15	1	1	1	1	14	1	1	1	0

wherein the column x_2 has been complemented and the columns x_1, x_3, x_4, and x'_2 become the 4th, 3rd, 2nd, and 1st columns respectively of Table I(b). The TRANSFORM transforms the function F to a new function G. It also changes the decimal designations of the minterms. Note that the least minterm of Table I(a), that is of F, is 3, and that of Table I(b), that is of G, is 5. This is the highest decimal designation of the least minterm that may be obtained. The TRANSFORM procedure ensures that the series of the linearization theorem is formed by a minimal number of terms. This in turn results in producing a minimal number of hidden units in the learnt representation. The transformed minterms are

stored in an array called the set of transformed minterms(STOM).

Now, we subject these transformed on-minterms to the REALIZE procedure, which starts with computing the first LS function of the series of G, that is, G_1. First, identify the unimplied minterms in the transformed tabular form. These terms are stored in an array called the set of unimplied minterms(SUM). The set of unimplied minterms, SUM, for G_1 is(see the IMS of Fig 1)

5 : 0 1 0 1
8 : 1 0 0 0

In Fig 1, the on-minterms of G are circled, and the UMs are in squares. As G_1 is a POLS function, its weights are all positive and ordered. Now, detect 10 combinations in the UMs. Its existence indicates inequalities among two successive weights. Hence 5 and 8 give the Weight Vector Inequalities as

$$w_4 > w_3 > w_2 = w_1$$

From this, let the weight vector variables be

c b a a

We now solve for the unknown weights **c,b,a** and the upper bound of threshold of G_1 by solving the required number of simultaneous equations. As soon as we know the ordered weight vector variables determined from the SUM, we know the exact number of independent equations required to find the unknowns of the realization vector, RV, namely the values of each of the weight variables and the upper bound of threshold T. Next we identify the border minterms, which are defined as follows.

Definition *5:* A minterm satisfying the following two relations is called a Border Minterm(BM). (a) It is neither directly nor distantly implied by any of the minterms of the SUM. (b) Each one of its directly implied minterms is either a UM or a minterm directly or distantly implied by at least one of the minterms of the SUM.

Fig.1 shows the two BMs **3** and **4** (in triangles) for the function G_1.

It is known from the property of a threshold gate that the weighted sum of some of the UMs will be equal to the upper bound of the threshold T, and the weighted sum of some of the BMs will be equal to the lower bound of threshold,t. We now make a conjecture that there always exists p number of UMs **each** of whose weighted sum is exactly equal to T, and q number of BMs each of whose weighted sum is **exactly equal to t, and p+q is greater** or equal to the exact number of independent equations required to solve for the weight vector variables and T. Again,in our algorithm the weights are minimum integers as can be obtained from the Dertouzous table[7]. Therefore, the minimum gap between two weights is always 1. Hence the gap between the lower bound and the upper bound of the threshold, T - t = 1, that is, t=T-1.

Definition *6:* Unimplied and border minterms whose weighted sums are exactly equal to the upper and lower bounds respectively of threshold, will be called distinguished **UMs(DUM),** and distinguished **BMs(DBM).** The set of DUM and DBM **are** found by an exhaustive search among the UMs and BMs whose number is usually not

very large. For function G_1 the DUMs 5 and 8 and the DBMs 3 and 4 give the four independent realization vector equations (RVE)

$$\begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 0 & -1 \\ 0 & 2 & -1 & -1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} c \\ b \\ a \\ T \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ -1 \end{bmatrix} \quad \text{yielding the solution} \quad \begin{bmatrix} c \\ b \\ a \\ T \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 3 \end{bmatrix}$$

As the gap between T and t is always 1, the actual threshold is fixed at the middle of the gap, that is, $\Theta = T - 0.5$. Consequently, the RV becomes, (< 3 2 1 1 > 2.5)

After RV has been computed, all the minterms greater than the least minterm are tested if they are realized by the realization vector(RV). During this process if any minterm which is not an on-minterm of the transformed function (circled in the IMS of Fig 1) is realized, then it is stored in an array called the set of error minterms(SEM). For G_1 this SEM turns out to be

SEM (G_1) = (7, 9, 10) shown by *1s in Fig 1.

Note that the checking for the realization need not go beyond 12(1100), since in this minterm all its 1s form a single string of consecutive 1s at the extreme left. It is obvious that when such a minterm is realized, any minterm which is larger than this will also be realized. Clearly, the function which has all the starting on-minterms and also the set of error minterms is an LS function and is the first function of the series, G_1.

$$G_1 = \sum (5\text{-}15)$$

Therefore, the realization vector, RV, of G_1 is given by

(< 3 2 1 1 > 2.5)

Note that while finding the LS function G_1 we had to add the set of error minterms to the set of on-minterms of G_1. These, therefore, must be subtracted and we must find the second function of the series, G_2. For this the starting set of minterms is the SEM(G_1). This becomes the SOM for the function G_2. These are once more transformed and stored in STOM only if the new transform obtains a larger designation of the least minterm. Otherwise, the previous transform is not altered. Following the same procedure as for G_1, the RV for G_2 is found. This turns out to be (<2 1 1 1>2.5). Then the SEM(G_2) is found. Here it becomes {11-15}. To cancel the effect of the newly found SEM, we compute another LS function G_3 whose on-minterms are the SEM(G_2). The RV for G_3 is determined to be (<3 2 1 1 >3.5). The SEM(G_3) however turns out to be the null set. Here, the REALIZE procedure terminates.

The weights and the thresholds of the functions G_1, G_2 and G_3 have been computed by the REALIZE procedure. The RESTORE procedure now assigns these weights with proper signs to the input variables, and also calculates the new threshold, following the standard methods given in [2,8]. The new weights and threshold give us the three LS functions F_1, F_2 and F_3, the three terms of the linearization theorem. These are

$$F(x_4,x_3,x_2,x_1) = F_1(x_4,x_3,x_2,x_1) - F_2(x_4,x_3,x_2,x_1) + F_3(x_4,x_3,x_2,x_1)$$

= (<1 2 -1 3>1.5) - (<1 1 -1 2>1.5)
+ (<1 2 -1 3>3.5)

Since F_1 and F_3 are excitory, they are connected to the output unit by height 1. F_2 is inhibitory and is therefore connected to the output unit by weight -1. The threshold of the output unit is always 0.5. The representation learnt is shown in Fig 2.

4 THE SINGLE PERCEPTRON SOLUTION

It is obvious that if the algorithm terminates by finding only one function of the Linearization theorem, then the given function is itself an LS function. The algorithm now declares the single hidden unit as the output unit. The representation so obtained is the single perceptron solution. For example, Fig 3 shows the single perceptron solution found by the IMS algorithm for example 4 of Brady et al[9]. This problem is interesting as it has been shown in [9] that back propagation fails to separate due to the presence of a 'spoiler' vector. Here, our algorithm has implemented the Boolean function

$F(x_3,x_2,x_1)=\sum(1,6,7)+\phi\sum(0,3,5)$.

where 0, 3, and 5 are the don't care or dc-minterms.

5 CONCLUSIONS

The algorithm as presented consists essentially of 3 procedures, TRANSFORM, REALIZE, and RESTORE. All these three procedures are non-iterative. Hence, the algorithm will be much faster than the BP. Unlike BP, the problem of local minima is non-existent in our algorithm. Therefore, the algorithm is much faster and will always produce the correct solution. The weights are always positive and negative integers(including zero), and like the Dertouzous Vector [7], they are also minimum integers. As we have shown in section 4 the same algorithm also produces the single perceptron solution, and there is no need for any exclusive perceptron training algorithm [10]. We have observed that there are many problems where the BP takes hours to train the network and even then may not produce the correct representation for its being trapped in local minima, our algorithm takes only seconds to produce the correct solution in a failproof manner. We hope that with further research and improvement, the IMS algorithm will ultimately replace Back Propagation totally in all artificial neural networks which are Boolean.

REFERENCES

[1] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, in Parallel Distributed Processing : Explorations in the Microstructures of cognition (Eds. Rumelhart and McClelland), Cambridge, Mass, MIT Press, Vol.1, pp.318-362, 1986.

[2] P.M. Lewis and C.L. Coates, Threshold Logic, New York, Wiley, 1967.

[3] N.N. Biswas, Introduction to Logic and Switching Theory, New York, Gordon and Breach, 1975.

[4] N.N. Biswas and R. Kumar, "A new algorithm for learning representations in Boolean neural networks", Current Science, Vol.59, No.12, pp.595-600, 25 June 1990.

[5] A.K. Sarje and N.N. Biswas, "Testing threshold functions using implied minterm structure", International Journal of Systems Science, Vo1.14, pp.497-512, 1983.
[6] H.C. Torng, "An approach for the realization of linearly separable switching functions", IEEE Trans of Electronic Computers, Vol.EC-15, pp.14-20, 1966.
[7] M.L Dertouzous, Threshold Logic: A synthesis approach, Research Monograph No. 32, Cambridge, MA, USA, The MIT Press 1965.
[8] Z. Kohavi, Switching and Finite Automata Theory, 2nd Ed., New York, McGraw-Hill, 1978.
[9] M.L.Brady, R.Raghavan, and J.slawny, "Back Propagation fails to separate where perceptrons succeed" IEEE Trans on Circuits and Systems", Vo1.36, No.5, pp 665-674, May 1989.
[10] P.D.Wasserman Neural Computing: Theory and Practice, New York, Van Nostrand Reinhold, 1989.

FIGURES

Fig.1

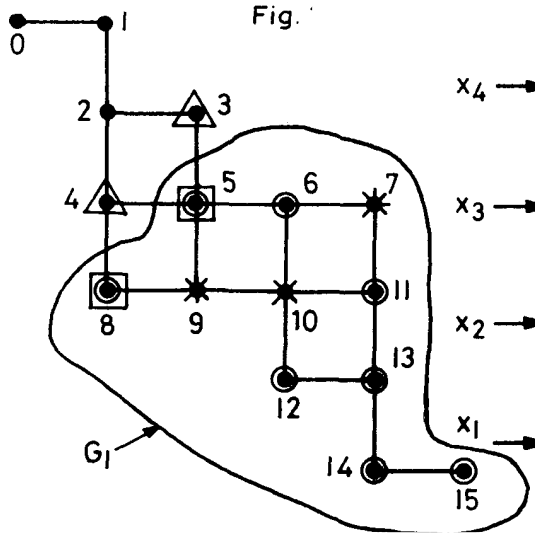
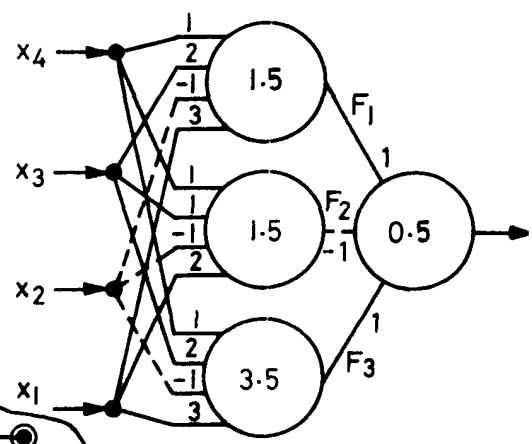


Fig.2



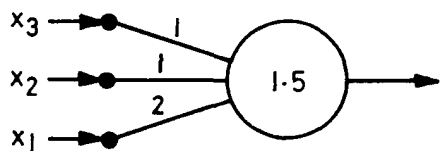


Fig. 3