On Construction of Binary Higher Order Neural Networks and Realization of Boolean Functions

Poorva Agrawal
Computer Engineering Department,
SGSITS, Indore

Aruna Tiwari
Computer Engineering Department,
SGSITS, Indore

ABSTRACT
We propose an Improved Binary Product Unit Neural Network (IBPUNN) and Improved Binary Pi-Sigma Neural Network (IBPSNN) with reduced number of hidden nodes. It is based on Binary Product Unit Neural Network (BPUNN) and Binary Pi- Sigma Neural Network (BPSNN) proposed by authors [1]. In this, the number of hidden nodes is equal to truth and false assignments respectively i.e. at most 2^{N-1}. In IBPUNN and IBPSNN the number of hidden nodes are further reduced if the inputs are at hamming distance one. In this case at the most 2^{N-2} hidden nodes are required. The weights used in these networks are 1, -1 and 0 only. The algorithms developed using these networks are constructive algorithm.

Keywords
Improved binary product unit neural network, Improved binary pi-sigma neural network, Boolean function, principle conjunctive normal form, principle disjunctive normal form , hamming distance.

1. INTRODUCTION
Recently, rapid advancement have made in neural networks. Neural networks have high acceptance ability for noisy data, high accuracy, low error rate and are therefore preferable in data mining [2]. Data mining is the process of analyzing data from different perspectives and summarizing it into useful information. Neural networks are widely applied in the solutions of several real world problems. Neural network method is used for classification, clustering [2], feature mining and pattern recognition. Watanabe [12] defines a pattern as an entity, vaguely defined, that could be given a name. For example, a pattern could be a fingerprint image, a handwritten cursive word, a human face. Many application domains such as data mining, image segmentation and handwriting recognition widely used pattern recognition techniques.

Neural networks have been successfully applied in a wide range of supervised and unsupervised learning applications. In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then calculated, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked. In Unsupervised learning techniques, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. The real world problem have large amount of data, the training algorithms suffer long training times and often reach local optima. Therefore higher order neural networks are used with joint activation terms relieving the network of the task of learning the relationships between the inputs [4].

1.1 Basics Concepts
The higher order neural network proposed in the paper is based on the Boolean functions. The input to the training of the neural network is binary data and that too in the form of truth tables. Also the samples are verified whether they are at hamming distance or not. In Binary Product-unit Neural Network (BPUNN) and Binary Pi- Sigma Neural Network (BPSNN) the weights are decided at once that is they require one-step training [1]. The number of hidden nodes is also reduced. Depending on the output of the truth table the network is formed.

2. SELECT WHETHER TO USE IBPUNN OR IBPSNN
We use IBPUNN (resp. IBPSNN) to implement a Boolean function of N variables with truth and false assignments. The hidden node number is equal to the number of truth (resp. false) assignments for Boolean functions if only the input samples are not at hamming distance 1, otherwise the number of hidden nodes get reduced. For a given complete truth table, we can choose a network with smaller size from IBPUNN or IBPSNN. For example, for a truth table of N variables with both truth and false assignments, if the number of truth assignments is less than that of false assignments in the truth table, we choose IBPUNN to implement the Boolean function with at most 2^{N-1} hidden node, and on the basis of input samples, i.e. if any two input samples are at hamming distance 1 then those two inputs will have 1 hidden node. Likewise this is for all those samples which are at hamming distance '1'. Contrarily IBPSNN is a better choice when number of false assignments is less than that of truth ones, and then hidden nodes are dependent of the type of the input samples whether they are at hamming distance 1 or not and accordingly the number of hidden nodes gets reduced.

2.1 IBPUNN
IBPUNN is based on BPUNN which uses the concept of product unit $\prod^{N}_{n=1} x_n^{w_n}$ instead of the traditional summation units $\sum^{N}_{n=1} x_n w_n$. A single hidden node is represented by H (w, x) which is the output of the binary product unit (BPU). The weight vector is represented as $w = (w_1, w_2, ..., w_N) \in \{-1, 1, 0\}^N$ and the input vector $x = (x_1, x_2, ..., x_N) \in Z_2^N$. It is represented in Figure 1.

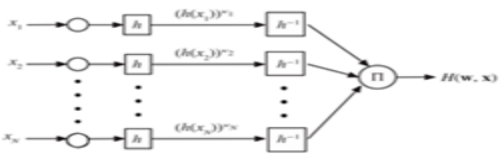

Fig 1: Structure of a Binary Product Unit

$$H(\mathbf{w},\mathbf{x}) = \prod_{n=1}^{N} h^{-1}\left((h(x_n))^{w_n}\right) \quad (2.1)$$

For considering the samples which are at hamming distance '1'. When w_n=1or -1

where $h(t) : Z_2 \rightarrow \{(1/2), 2\}$ is defined by

$$h(t) = \begin{cases} 2, & t = 1; \\ \frac{1}{2}, & t = 0 \end{cases} \quad (2.2)$$

and $h^{-1}(t)$ is the inverse function of $h(t)$

$$h^{-1}(t) = \begin{cases} 1, & t = 2; \\ 0, & t = \frac{1}{2}. \end{cases} \quad (2.3)$$

And when w_n=0 then,

$(h(x_n))^{w_n}$ Will always be 1

Therefore, $h^{-1}(t) = 1, t = 1$ (2.4)

The output of a IBPUNN is represented by y, which have N inputs and number of hidden nodes is $1 \le P \le J$ where J is the number of truth assignments and P is the number of hidden nodes and weight which is denoted by $w_p = w_{p,1}, w_{p,2}, \ldots\ldots, w_{p,N} \in -1, 1, 0$ N and hidden to output weight are fixed to 1. Thus the final output of IBPUNN is

$$y = g\left(\sum_{p=1}^{P} H(w_p, x)\right) \quad (2.5)$$

$$= g\left(\sum_{p=1}^{P}\left(\prod_{n=1}^{N} h^{-1}(h(x_n))^{w_{p,n}}\right)\right) \quad (2.6)$$

Where H (w_p, x) stands for output of p^{th} hidden node and

$$g(t) = \begin{cases} 1, t > 0 \\ 0, t = 0 \end{cases} \quad (2.7)$$

The value of P varies from 1 to J, but on the basis of hamming distance if all the samples are at hamming distance'1', then the number of hidden nodes get reduced to 2^{N-2}, whereas in BPUNN the number of hidden nodes will be J, where value of J ranges from 1 to 2^{N-1}. IBPUNN is represented in Figure 2.

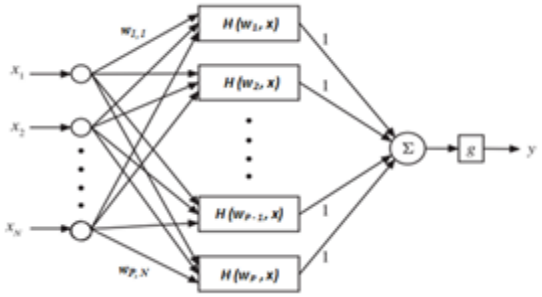

Fig 2: Structure of N-P-1 IBPUNN

2.2 IBPSNN

IBPSNN is based on the BPSNN where y is the output of the network, and the weight vector is represented as $v = v_1, \ldots\ldots, v_N \in 1, -1, 0^N$ and the input vector $x = (x_1, x_2, \ldots\ldots, x_N) \in Z_2^N$.

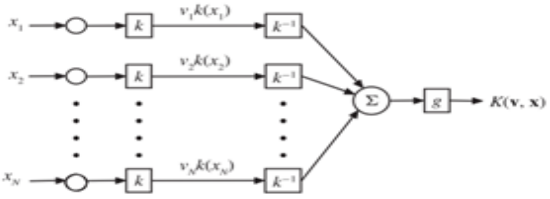

Fig 3: Structure of a hidden node of a IBPSNN

The output of a single hidden node is

$$k(v,x) = g\left(\sum_{n-1}^{N} k^{-1}(v_n k(x_n))\right) \quad (2.8)$$

When v_n=1 or -1 then, Where g(t) is

$$g(t) = \begin{cases} 1, t > 0; \\ 0, t = 0; \end{cases} \quad (2.9)$$

K (t): $Z_2 \longrightarrow$ {-1, 1} is defined by

$$K(t) = \begin{cases} 1, t = 1; \\ -1, t = 0; \end{cases} \quad (2.10)$$

And k^{-1}(t) is the inverse function of k (t)

$$K^{-1}(t) = \begin{cases} 1, t = 1; \\ 0, t = -1; \end{cases} \quad (2.11)$$

And when v_n=0 then,
$v_n k(x_n)$ Will always be 0
Therefore, $K^{-1}(t) = 0, t = 0;$ (2.12)

The number of hidden nodes is Q, where $1 \le Q \le I$, I is the number of false assignments and Q varies from 1 to I, the Q is based on the input samples. If in BPSNN ,the no of hidden nodes are I which can be almost 2^{N-1}, if the input samples are at hamming distance 1, then the number of hidden nodes get reduced and it become 2^{N-2}.

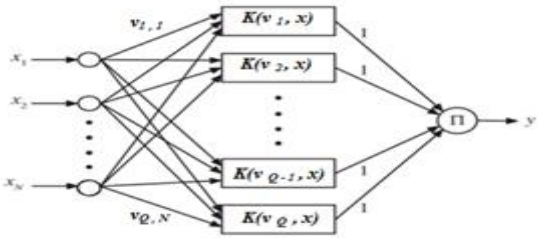

Fig 4: Structure of a N-Q-1 IBPSNN

3. LEARNING ALGORITHM

Step1. The data is converted in the binary form so that it can be directly made as an input for training the neural network.
Step2. The input data is the binary form with a 2 class problem with 1 and 0 as output. The count of number of samples for each category is done.
Step3. If number of samples of truth value is less than the number of samples of false values then training of IBPUNN is done else training of IBPSNN is done.
Step4. If any of the samples are at hamming distance 1 then a single weight is generated for those 2 samples. Otherwise there will be 2 separate weights for each sample.
Step5. Comparison of this proposed neural network with the existing network that is BPUNN and BPSNN is done on the basis of number of hidden nodes.

4. ILLUSTRATIVE EXAMPLES

Let us consider a set of three samples. These samples are taken with three inputs so as to compare the hidden nodes of IBPUNN with BPUNN. Therefore the samples in this case will only belong to the category of truth output value.

4.1 Example 1

Table 1. Sample 1

x_1	x_2	x_3	Y
0	0	0	1
1	0	0	1
0	1	1	1

Step1. As the truth table is of three inputs that is total combinations are eight. Out of which 3 samples are of output 1 and other five are of sample 0.
Step2. Number of samples of truth value is less than the number of samples of false values then training is done through IBPUNN.
Step3. In IBPUNN, the first and second samples are at hamming distance '1'. Therefore, the number of hidden nodes will only be 2 as for the first and second sample we will have common weight that is 0,-1,-1 and for the third sample the weight will be -1,1,1.
Step4. In the case of BPUNN, the numbers of truth assignments are 3, so there will be 3 hidden nodes. The weights corresponding to these hidden nodes are -1,-1,-1, 1,-1,-1 and -1, 1, 1,

4.2 Example 2

Table 2. Sample 2

x_1	x_2	x_3	Y
1	1	1	1
0	1	0	1
1	0	0	1

Step1. As the truth table is of three inputs that is total combinations are eight. Out of which 3 samples are of output 1 and other five are of sample 0.
Step2. Number of samples of truth value is less than the number of samples of false values then training is done through IBPUNN.
Step3. In IBPUNN, as none of the samples are at hamming distance 1, therefore number of hidden nodes will be equal to number of truth assignments and here it is equal to 3.
Step4. In the case of BPUNN, the numbers of truth assignments are 3, so there will be 3 hidden nodes.

5. EXPERIMENTS AND RESULTS

In this section the proposed neural network IBPUNN and IBPSNN have been tested in MATLAB 7.8.0 (R2009a) on dataset and the results are obtained. The dataset used are Tic-Tac-Toe Endgame database, Contraceptive Method Choice Data Set, Car Evaluation, Haberman's Survival Data, whose values are in numeric form. These dataset are integer values therefore it has been pre-processed and finally binary values are obtained. The training phase now starts in which a particular network is chosen, there is a need to count the output of the input dataset which has been converted into binary form so as to find the number of samples in each category. The binary data is treated as a truth table and the output value will always be either '1' or '0'. The details of the datasets are shown in the table 3.

Table 3. Basic information about the dataset

Dataset	Samples	Features	Classes
Car Evaluation	1594	6	2
Haberman's Survival	306	4	2
Tic-Tac-Toe	958	9	2
Contraceptive Method Choice	642	10	2

The algorithm used to train the network can be applied on any neural network the system will be trained but to obtain the optimized network with least number of hidden nodes will be chosen. The Improved Binary Product-unit Neural Network (IBPUNN) is opted when the number of truth assignments is less than the number of false assignments and Improved Binary Pi- Sigma Neural Network (IBPSNN) when the number of false assignments is less than number of truth assignments. Accordingly the training and testing samples are found of all the datasets. This is represented in the table 4.

Table 4. Training and testing samples of datasets

Dataset	No. of Instances	No. of Instances in training	No. of Instances in testing
Car Evaluation	1594	384	1210
Haberman's Survival	306	81	225
Tic-Tac-Toe	958	332	626
Contraceptive Method Choice	642	227	415

The numbers of hidden nodes of the proposed neural network get reduced if the training samples of the datasets are at hamming distance '1'. However, the training time is bit increased as the weight matrix is made by comparing the samples so as find whether they are at hamming distance 1 or not. The datasets like Haberman's Survival, Car Evaluation dataset and Contraceptive Method Choice data sets have there samples at hamming distance '1'. The classification accuracy of the proposed neural networks remains same i.e. 100%. Results of IBPUSNN and IBPSNN are compared with BPUNN and BPSNN respectively. This comparison is represented in the table 5.

Table 5. Comparison of IBPUNN and IBPSNN with BPUNN and BPSNN

Dataset	No. of Hidden nodes in IBPUNN/ IBPSNN	No. of Hidden nodes in BPUNN/ BPSNN	Training time in IBPUNN/ IBPSNN	Training time in BPUNN/ BPSNN	Classification accuracy
Tic-Tac-Toe Endgame	332	332	2.6132	0.0036	100%
Haberman's Survival	77	81	0.3839	0.0002	100%
Car Evaluation	199	384	2.2185	0.0021	100%
Contraceptive Method Choice Data Set	202	227	1.3128	0.0021	100%

6. CONCLUSION

In this paper, we proposed two binary higher order neural networks: IBPUNN and IBPSNN. We have proved by various examples that these networks have reduced number of hidden nodes and they can realize Boolean functions. These networks are further compared with BPUNN and BPSNN and numerical experiments have been performed. The structures of IBPUNN and IBPSNN have been derived from BPUNN and BPSNN respectively. The original network structures were modified so as to reduce the number of hidden nodes in the network using the concept of hamming distance. The weights used in these structures are simply 1, -1, 0 by applying constructive training. Our approach deals with complete truth table of N variables with both truth and false assignments. The corresponding Boolean functions could be realized by accordingly choosing IBPUNN and IBPSNN such that at most 2^{N-1} hidden nodes are needed. If these samples are at hamming distance 1 then in such case at most 2^{N-2} hidden nodes will only be needed. The two exceptional Boolean functions with 2^N truth assignments but no false assignment, or with 2^N false assignments but no truth assignment, could be realized by IBPUNN or IBPSNN, respectively, with 2^N hidden nodes.

7. REFERENCES

[1] Chao Zhang, Jie Yang, and Wei Wu : Binary Higher Order Neural Networks for Realizing Boolean Functions, IEEE Transactions on Neural Networks, vol. 22, no. 5, pp. 701-713, May 2011.

[2] Xianjun Ni. : Research of Data Mining Based on Neural Networks, World Academy of Science, Engineering and Technology 39, pp 381-384, 2008.

[3] A.D Kulkarni and V.K Muniganti. : Fuzzy Neural Network Models for Clustering, pp 523-528.

[4] Adam Knowles, Abir Hussain , Wael El Deredy, Paulo G.J Lisboa and Christian Dunis. : Higher Order Neural Network with Bayesian Confidence Measure for Prediction of EUR/USD Exchange Rate".

[5] R. Johnsonbaugh, Discrete Mathematics, 5th ed. London, U.K.: Pearson, 2000.

[6] R.Durbin and D.E. Rumelhart. : Product units: A computationally powerful and biologically plausible extension to backpropogation networks, Neural Comput., vol. 1, no. 1, pp. 133-142, 1989.

[7] Y.Shin and J. Ghosh. : The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation, in Proc. Int. Joint Conf. Neural Netw., vol. 1. Seattle, A, Jul.1991, pp.13-18

[8] G. Fahner. : A higher order unit that performs arbitrary Boolean functions, in Proc. Int. Joint Conf. Neural Netw., vol. 3. San Diego, CA, Jun. 1990, pp. 193–197.

[9] J. Y. Li, T. W. S. Chow, and Y. L. Yu. : The estimation theory and optimization algorithm for the number of hidden units in the higher order feedforward neural network, in Proc. IEEE Int. Conf. Neural Netw., vol. 3. Perth, Australia, Nov.–Dec. 1995, pp. 1229–1233.

[10] D. Cheng and H. Qi. : State–space analysis of Boolean networks, IEEE Trans. Neural Netw., vol. 21, no. 4, pp. 584–594, Apr. 2010.

[11] M. L. Johnson. : Perceptron-how this neural network model lets you evaluate Boolean functions, IEEE Potent., vol. 12, no. 3, pp. 17–18, Oct. 1993.

[12] S. Watanabe. : Pattern Recognition: Human and Mechanical. New York: Wiley, 1985.