IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 1, Issue 6 (July-Aug. 2012), PP 36-42
www.iosrjournals.org

Design and Implementation of Binary Neural Network Classification Learning Algorithm

Shweta Kori[1], Satyawan Mehra[2],Pratik Jain[3]
Chameli Devi School of Engineering[1,3]
Shri GS Inst of Tech.& Sci.[2]
Indore (M.P.) India 452001

Abstract—*In this paper a Binary Neural Network Learning (BNN-CLA)[1] is analyzed and implemented for solving multi class problem normally the classifier are construct by combining the outputs of several binary ones. The BNNC offers high degree of parallelism in hidden layer formation for all multiple classes to reduce the time for learning. The learning method is an iterative process to optimize the classifier parameters. In this approach, overlapping problem is tackled to enhance the performance of classifier by changing hyper-sphere radius. Exhaustive testing is carried out. Accuracies and number of neuron are evaluated and compare with BNNC[4]. The method have been tested on Fisher's well known Iris data data set and experimental result shown the classification ability improved by using FCLA algorithm. While comparing with BNNC[4] in most cases accuracies improved in BNNL because of elimination of samples which are lying in overlapping region of classes. Thus tacking overlapping issue improved performance of this classifier.*
keyword: *Semi-Supervised classification, BNN Geometrical Expansion, Hyper sphere, overlapped classes.*

I. Introduction

In the last two decades, binary neural networks (BNNs) have attracted attention of many researchers and now there have been many established approaches for the construction of BNNs. They include BooleanLike Training Algorithm (BLTA)[3], Improved Expand and Truncated Learning (IETL)[8]. These learning for solving classification problem,two classes or multiclasses.in case of multi class overlapping of samples becomes a measure issue. Due to this accuracies of learning algorithm decreases. For solving the problems on binary neural networks, many concepts have been introduced for the optimization of the network structure as well as the Boolean function [10]. Gray and Michel's Boolean Like Training Algorithm (BLTA) [17]used the idea of Karnaugh map to simplify and decrease the complexity of neural network. This algorithm forms the structure of four layer feed forward network with the features of modifications and incremental learning. Modification involves the changing of learnt input/output relationship with new one without retraining the network. Xiaomin, Yixian, Zhang [9] introduced Boolean function scheme with three layer network and the concepts of hamming distance ball and cube coverage for simplification of the Boolean Function and the resultant network. Zhang at. el. [2], investigated three layer BNN by developing a deterministic algorithm called Set Covering Algorithm (SCA). It finds a family of subsets of various unit spheres in the hamming space (input space with the hamming distance between two inputs) which is required for covering all inputs having the same desired output. Using this concept, it determines all the hidden neurons. They proved that the number of hidden neurons required are much less than that generated by using a two parallel hyperplane method. They also presented a lower bound on the required number of hidden neurons by estimating the Vapnik-Chervonenkis dimension. Kim and Park [10] introduced a linearization technique which transforms a set of linearly inseparable binary patterns to a set of linearly separable ones. Using this technique Kim and Park have introduced a learning algorithm for discrete multilayer perceptrons for binary patterns which Yi Xu and Chaudhari [14] have presented the usefulness of this algorithm for multi-class classification. Sang-Kyu Sung and Jong Won Jung proposed Newly Expanded and Truncated Learning Algorithm (NETLA) [5] which reduces the number of hidden neurons and connections compared with ETL and BLTA. Wang and Chaudhari [3] have introduced learning algorithms for BNN named as Multi-Core Learning (MCL) algorithm and Multi-Core Expand-and-Truncate Learning (MCETL) algorithm. These algorithmsgive simpler equations for computing the various factors required for the training. The number of operations needed for these algorithms is less as compared to ETL and IETL. Learning Algorithm called Constructive Set Covering Learning Algorithm (CSCLA) was proposed to train the same kind of three layers BNN for the generation of arbitrary Boolean function by Xiaomin and Yixian [4]. Parekh andYang [6] proposed a constructive neural network learning algorithms for pattern classification. They proposed this method for handling multiple output categories and real valued pattern attributes. Gazula and Kabuka [8] have presented two supervised pattern classifiers designs using Boolean neural networks namely, nearest-to-an-exemplar and Boolean knearest neighbor classifier. Wang and Chaudhari [1] proposed a geometrical approach for the construction of binary neural networks called as Fast Covering Learning Algorithm

(FCLA). The method works for two class problems and always results in a three layer network structure. A learning algorithm called Constructive Semi-Supervised Classification Algorithm (CS-SCA) was proposed to train the same kind of three layer BNN for the generation of arbitrary Boolean function for classifying semi-labeled data by Chandel, Tiwari and Chaudhari [19]. In this paper a Implementation of binary neural network classifier Algorithm is proposed in which, the hidden layer training of multiple classes is being done in parallel. During hidden layer training, the samples lying in overlap region of various classes are also checked to improve the accuracy. This paper is divided into six sections. Section II describes the preliminary theory for establishing the algorithm. Section III is presented with the proposed learning algorithm. Section IV is presented with experiments conducted with few benchmark datasets and finally, section V is containing concluding remarks for the work.

II. Overview Of Bnnc

A. Geometrical Basic In FCLA[1], a neuron is visualized as an hyper sphere.

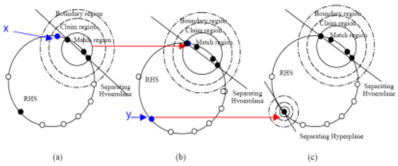

Figure 1. Geometrical Basic

It reduces the number of hidden neurons by expanding the scope of each hidden neuron in the training process based on multi-level geometrical analysis. It handles two class problems. For solving multiclass problem, multiple hyper spheres are evaluated corresponding to multiple classes. Suppose that there are K classes G1, G2… Gk are to be given. For each of the K classes, a set of ‘p’ number of n-bit training inputs are given. During the expansion of hyper spheres belonging to different classes, overlap may occur. The algorithm proposed herein handles multiclass problems by taking care of the samples lying in such overlapped regions. Geometrical expansion of a neuron is being done by considering three radii, r1, r2, r3 with the same center C of the hyper sphere. Radius r1 denotes the minimum value such that vertices are exactly in or on the hyper sphere. Thus, the expansion region formed for this is called as ‘match region’. Radius r2 is being used for further expansion (immediate expansion) of hyper sphere. Thus, a new vertex is being added in this expanded region and is called to be added in the ‘claim region’. Radius r3 defines the third region called as ‘boundary region’. A new vertex cannot be added in this region, but is expected to be included in the next (or later) construction iteration. These radii must satisfy the condition r1<r2<r3. Three center and radii are defined as follows:
Where $□_1$ and $□_2$are the parameters for geometrical expansion. $□_1$ is the Hamming distance of the vertices in the claim region from the match region, and $□_2$□ is the Hamming distance of the vertices in the claim region from the boundary region. Parameters for the Hidden Neuron in the form of a hypersphere lab are given as follows. Centre of ith hyper sphere is given as:

The centre is defined as follows:

$$c_i = \sum_{k=1}^{v} \frac{x_i^k}{v} \qquad (1)$$

***Three radii are defined as follows*:**

$$r_1^2 = \max_{k=1}^{v} \sum_{i=1}^{n} \left(x_i^k - c_i\right)^2 \qquad (2)$$

$$r_2^2 = r_1^2 + 1 \qquad (3)$$

$$r_3^2 = r_2^2 + 1 \qquad (4)$$

where*jpx* represent the jth bit of pth input vertex and
v represents the number of vertices added in a neuron. Integer valued weights for the neuron are:

Threshold t1 corresponding to r1 is

$$w_j = 2\sum_{p=1}^{v} x_p^j - v \qquad (5)$$

Threshold t1 corresponding to r1 is

$$t_1 = \min_{k=1}^{v} \sum_{i=1}^{n} w_i x_i^k \qquad (6)$$

Threshold t2 corresponding to r2 is

t1 $= t_1 - v□_1^2$ (7)

Threshold t3 corresponding to r3 is

t3 $= t_1 - 2v□_1^2$ (8)

Network Architecture:

for a given k- class problem {G1,G2,.....GK} for each and every class,we separately apply hidden layer training of the algorithms proposed next partial network formed is visualized as follows:

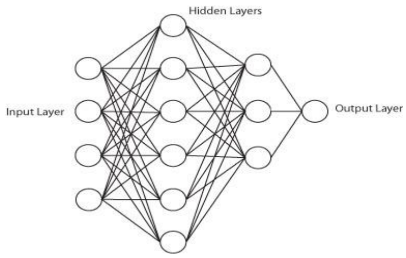

Figure2. Partial network structure for hidden layer. Prepare Your Paper before Styling

III. TRAINING FOR THE CONSTRUCTION OF NETWORK

For our extension, there are two broad steps involved in the construction of network:

A. Training of hidden layer: The training of hidden layer is done in parallel for each of k classes using FCLA[2] as follows:

Algorithm 1

1. For a given class Ck, take set of true vertices (x1,x2...xm), each vertex is n-bit long represented as xij , where $1 \leq j \leq n$.

2. For each of the input data-

For i=1 to m do
Begin
if (i=1) then
-add a new neuron with respect to this input(xi) therefore evaluate following parameters--Center C (using equation (1))
-Radius r1, r2 ,r3 (using equations (2), (3), (4))
-Weights (w1 , w2 ,... wn) represented as weight vector W (using equations (5))
-Thresholds(t1,t2,t3) (using equations(6), (7), (8))
else
begin
-check this input data(xi) with respect to the existing neurons
-for each of the pth neuron do the following checks

<Cond1> if(Wxi >= t1) then

-this input is already covered by the pth neuron so simply exit & take next input(match region)

<Cond2> if(t2 <= Wxi <=t1)

-input data is within the claim region
-update the parameters of pth neuron by using the formulae in section 3
-center C, radius, weights, threshold
-exit & take next input

<Cond3> if(t3 > Wxi)

-if this condition is true for all the neurons then a new neuron is being added.
-Evaluating all the parameters center, radius, weight & thresholds in section 3

<Cond4> if(t3<=Wxi < t2)

-the vertex is within the boundary region
of the neuron, so we first
-examine whether other available neurons
can claim it?
-if it can not be included in any other
available neuron, we "put aside" for
reconsideration after other vertices are
processed.
-inclusion of other vertices to existing
neurons results in the expansion of "match" & "claim" regions of the neurons;
other vertices "putaside" may be claimed.

<Cond1> & <Cond2> is being retested.
End else
End for 1

3. Modification process: Apply all vertices belonging to other classes (say, false vertices) to the hidden layer neurons trained for a class. If the output is zero then
omit it. If output is one then we will represent the wrongly represented vertices by additional hidden neurons by applying step 2.

4. Repeat steps 2 and 3 for each of the class.

5. Stop.

Overlap Test: The learning algorithm allows overlap ofhyperspheres from the same class and eliminates overlap between hyperspheres from separate classes.
Overlap test is performed as soon as hypersphere is
Expanded by case II or created in case III. Overlap test is carried out as follows:

(i) Overlap test for Cond2 II: Let hypersphere Ha is expanded to include the input pattern Xi and expansion of hyper sphere Hb has created a overlap
Which belongs to other class? Suppose C1 and r1, r2
Represent centre and radii of the expanded hypersphere Ha, C2 and, *r1*,r2 are centre and radii of the hypersphere of other class. Then if

$$\sum_{j=1}^{n}(\mathbf{c1i - c21)2)1\backslash 2 \leq r1 + r'1} \qquad (11)$$

OR,

$$\sum_{j=1}^{n}(\mathbf{c1i - c21)2)1\backslash 2 \leq r2 + r'2} \qquad (12)$$

Means hypersphere form separate classes are overlapping.

(ii) Overlap test for Cond2 III: If the created hypersphere falls inside the hypersphere of other class means there is an overlap. Suppose hypersphere Ha is created to include the input pattern Xi and hypersphere Hb belongs to other class. The presence of overlap in this case can be verified as follows:

$$\sum_{j=1}^{n}(\mathbf{c_{1i} - c_{21})^2)^{1\backslash 2} \leq r_1} \qquad (13)$$

Removing Overlap:
If equation (11) or (12) is satisfied for any twohypersphere from different classes then there is a overlap.The overlap in either of the situation is removed byrestoring the radius of just expanded hypersphere. Let Hbbe the expanded hypersphere then it is contracted as:

$$\mathbf{r_1\,new = r_1} \qquad \textbf{if con. (11)true}$$

else

$$\mathbf{r_2\,new = r_2\,old} \qquad \textbf{if con.(12)true}$$

and a new hypersphere is created for the input pattern
Described by equations (1) to (4).
For step (ii) the overlap can be eliminated as follows:

$$\mathbf{r_1\,new = \sum_{j=1}^{n}(c_{1i} - c_{21})^2)^{1\backslash 2} - \delta_1}$$

and

$$\mathbf{r_2\,new = \sum_{j=1}^{n}(c_{1i} - c_{21})^2)^{1\backslash 2} - \delta_2}$$

Where δ_1 and δ_2 are small number. Selected just enough to remove the overlap. In our experiments the values of $\delta 1$ and δ_2 are chosen between 0 and 1. Hence hypersphere Ha is contracted just enough to remove the overlap.

IV. Training for output layer:

We take (log K) neurons at the output layer to represent all K classes. Threshold values corresponding to all the neurons are set to one. For deciding the weight values, we choose a number 'np' which gives sufficient bit combination for representing the given number of classes. In Binary neural networks, the outputs are also in the binary form. Therefore, the number of bits and bit combinations are to be decided for representing the different given classes. For example, for a 3-class problem, the nearest integer, which contains all the
multiples of 2 is 4. This could be written as 22=2np. Thus the combinations of 2-bits(np) gives the 4-possibilities and a 3-class, 4-class problem can be represented by the combination of 2 bits. Thus for a given K-class problem, the numbers of bits are decided as follows:
(i) Take a nearest integer says 'b' such as it contains all the multiples of 2. b>=K
(ii) Rewrite 'b' in the form of 2np, where 'np'
represents the power of 2.b = 2np

Thus 'np' gives the number of bits required to represent a given K-class problem. After deciding the number of bits for representing the classes, a table is formed containing bit combinations corresponding to each class. Each column of this table is mapped to output neurons. Thus the table contains log(K) columns corresponding to output neurons.

Table I. Bit Combinations For 3-Class Problem

S.No.	**Classes/No. of Output Neuron**	**O1**	**O2**
	G1	1	1
	G2	1	0
	G3	0	1

Columns of the table represent the weight values of a Output neuron corresponding to the different hidden neurons of different classes.

V. Experimental Work

The computational experiments were carried out on a Pentium II, 512 MB RAM, 700 MHz using the code.
Implemented in Matlab tool. For each data set, 75% samples are used for training and 25% samples are used for testing. The data values of datasets are transformed to have zero mean and one variance. These transformed data samples are then converted into binary values as follows: we requantize the data into eight levels. These eight levels are then binary coded using three bits each.

Step-1: Input the data file
Step-2: Apply each sample as an input *u* in quantized function given in step 3.
Step-3: Quantized value can be obtained by using the function: *y* □ □*uencode* (*u*, *n*, *v*)
Step-4: Repeat the step 3, till you don't get the final binary coded sample.
Values of δ_1 □and □ δ_2 are chosen as 0.052 and 0.065 respectively for conducting all experiments

TABLE II. DATASETS DESCRIPTION

Dataset	Number of Features	Number of Classes	Total Samples
Iris	4	3	150
Glass	13	7	214
Riplay	2	2	1000
Car	6	4	1728
WBC	9	2	527
Ecoil	7	8	336

TABLE III. NUMBER OF HIDDEN NEURONS EVALUATED EXPERIMENTALLY

Dataset	$\sigma_{1=1,}$ σ_2=0.4	σ_{1}= σ_2
Iris	5	6
Glass	20	15
Riplay	4 8	49
Car	70	56
WBC	70	59
Ecoil	10	6

TABLE IV. TIME (IN SEC.) REQUIRED FOR TRAINING EXPERIMENTALLY

Dataset	σ_1=1,σ_2=0.4	$\sigma_{1=}$,σ_2 =
Iris	0.123	0.987
Glass	0.438	0.561
Riplay	1.932	1.893
Wine	0.541	0.543
WBC	1.444	1.871
Vehicle	0.612	0.916

TABLE V. % ACCURACIES FOR DIFFEREN DATASETS

Dataset	σ_1 =1,σ_2	$\sigma_{1=,}$,σ_2=
Iris	93.00	88.66
Glass	72.61	63.00
Riplay	83.00	81.29
Wine	81.89	62.12
WBC	84.15	79.59
Vehicle	67.32	69.32

The results for number of neurons and computing time of training and accuracies in percentage are given in table III and table IV and table V respectively. These results for the datasets Glass and WBC are then compared with [16]. Our methods use much less CPU time than [16] method. The numbers of neurons required are less in case of WBC datasets.

Accuracies are also comparable. Note that the addition of every new hyper plane for a class alters the positioning of previous hyper planes and gradually increase the separation from the other classes thus maximizing the linear independency. If overlap occurs then our algorithm handles it by varying the values of δ_1 and δ_2 □ . By varying the values of δ_1 and $\delta_{2,}$ good accuracies can be achieved in any Application.

IV. CONCLUSION

In this paper, we extend FCLA method for multiclass problems by designing classifiers using coding schemes. The hidden layer layer trained is in modular form. Thus modules in the hidden layer corresponding to each reduces training time. BNN-LCA is tested with various datasets. We observed that the training time of the proposed method is also comparable with []. An analytical formulation for number of hidden neurons are derived which is giving the result of the O(log(N)), Where N represents the number of inputs.s

References

1. A. Tiwari, and N.S. Chaudhari, �Design of output codes for Fast Covering Learning using basic Decomposition Techniques,� *Journal of Computer Science*, (Science Publications, NY, USA), Vol. 2, No. 7, pp. 565-571 (July 2006).
2. D.Wang and N.S.Choudhari, "A novel training algorithm for Boolean neural networks based on multi level geometrical expansion," *Neurocomputing* 57C (2004) 455-461
3. Narendra S. Chaudhari and Aruna Tiwari ,"Binary Neural Network Classifier and it's Bound for the Number of Hidden Layer Neurons", ICARCV2010 978-1-4244-7815-6/10/$26.002010 IEEE
4. Di Wang, Narendra S. Chaudhari and Jagdish Chandra Patra, "Fast Constructive-Covering Approach for Neural Networks," *School of Computer Engineering, Nanyang Technological University,* Singapore,
5. J.H.Kim and S.K.Park, "The geometrical learning of binary neural neworks," *IEEE Transaction. Neural Networks* 6(1995) 237-247,
6. S. Gazula, and M. R. Kabuka, "Design of supervised classifiers using Boolean neural networks," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol.17, No. 12, IEEE, USA, pp.1239-1246, Dec 1995.
7. Arvind Chandel, A. Tiwari, and N. S. Chaudhari, "Constructive Semi-Supervised Classification Algorithm and its Implement in Data Mining", Third International Conference on Pattern Recognition and Machine Intelligence (PReMI' 2009) (published in Lecture Notes in Computer Science, vol. no. 5909, ISBN 978-3-642-11163-1, SpringerVerlag Berlin Heidelberg) , IIT Delhi, India Dec. 2009.
8. Dan Zhang, Jingdong Wang, Fei Wang and Changshui Zhang, "Semi-Supervised Classification with Universum," *State Key Laboratory on Intelligent Technology and Systems, Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University*, Beijing, 100084,

9. Di Wanga and Narendra. S. Chaudhari, "A constructive unsupervised learning algorithm for clustering binary patterns," *in: Proceeding of international joint conference on Neural Networks 2004*, vol.4, Budapest, Hungary, 2004, pp. 1381-1386,
10. Kishan Mehrotra, Chilukuri K. Mohan and Sanjay Ranka, 1997. Elements of Artificial Neural Networks : Cambridge, MA:MIT Press.
11. Thomas G. Dietterich, Ghulum. Bakiri,1995. Solving Multiclass Learning Problems via Error-Correcting Output Codes : Journal of Artificial Intelligence Research, Vol. 2 : 263-286.
12. Donald L. Gray and Anthony N. Michel, 1992. A training algorithm for binary feedforward neural networks. IEEE Trans : Neural Networks, Vol. 3,No. 2, IEEE, USA, pp :176-194