Journal of Computer Science 2 (7): 565-571, 2006
ISSN 1549-3636

Design of Output Codes for Fast Covering Learning using Basic Decomposition Techniques

Aruna Tiwari[1] and Narendra S. Chaudhari[2],
[1]Faculty of Computer Engineering Department, Shri G. S. Institute of Technology & Science (SGSITS), 23, Park Road, Indore 452003 (M.P.) INDIA
[2] Faculty of School of Computer Engineering(SCE), Nanyang Technological University(NTU), 50,Nanyang Avenue, Nanyang Technological University(NTU), Singapore 639798 SINGAPORE

Abstract: We propose the design of output codes for solving the classification problem in Fast Covering Learning Algorithm (FCLA). For a complex multi-class problem normally the classifiers are constructed by combining the outputs of several binary ones. In this paper, we use the basic methods of decomposition; one per class (OPC) and Error Correcting Output Code (ECOC) with FCLA, binary to binary mapping algorithm as a base binary learner. The methods have been tested on Fisher's well-known Iris data set and experimental results show that the classification ability is improved by using ECOC method.

Key words: Binary neural network , One per class, Error correcting output code.

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 Boolean Like Training Algorithm (BLTA)[3], Improved Expand and Truncated Learning (IETL)[8]. In these methods, predefined output codes are used for the representation of multiple classes. Using predefined output codes makes the problem independent of the specific application and class of hypotheses used to construct binary classifiers[9]. Experimental work has shown that output coding can greatly improve various performance parameters like generalization, prediction accuracy[1] etc. Several output coding methods have been suggested and tested so far, such as comparing each class against the rest (One Per Class: OPC), comparing all pairs of classes (Pair Wise Coupling: PWC), random codes, exhaustive codes, Error Correcting Output Codes, Margin Classfiers[1,5,6,7].

In this paper, we extend Fast Covering Learning Algorithm (FCLA)[2] for multi-class problem (i.e., K-classes, where K>2). Further, this paper addresses the design of output codes for a binary to binary mapping learning. In our work, we use two output coding schemes One-Per-Class (OPC) and Error Correcting Output Code (ECOC). Output Coding of multi-class problems is composed of two stages. In the training stage , we need to construct hidden layer by independent K binary classifiers where K is the number of classes to be learned. The output layer is then constructed by training of number of neurons as per the coding scheme used. In the second stage, the classification part, the applied sample is predicted by combining various binary classifiers. OPC separates one class from all other classes and ECOC consists of several dichotomizers with class redundancy to get robustness in case some dichotomizers fail[5,6,7]. ECOC approach improves the generalization performance[1,5,7]. These coding schemes are used for output coding for the training phase of the neural network. In the reconstruction stage, when new samples come, some similarity measure is required to find out the class to which it belongs, if the generated string is in binary form, the hamming distance criteria is being used for deciding the class to which new sample belongs[5,7].

In case of OPC, for the training of output layer, a class is separated from the rest of the classes. Therefore, at the output layer, a single neuron per dichotomizer is taken to collect the outputs from the hidden layer neurons of their respective class. The weights and thresholds in the output layer are set to one for each of the dichotomizer/neuron.

In ECOC[1], each class is assigned a unique binary string. We refer to these strings as codewords. Then we train K classifiers at the hidden layer and l number of output neurons at the output layer (where l is the length of the codeword). The predicted class is one whose codeword is closest to the output generated. The similarity measure is the Hamming distance ; (i.e., the number of bits different from the codeword bits).

We show that the use of ECOC method for FCLA improves the generalization capabilities over the OPC. This comparison has been tested by experimenting on Iris data set. Also, utilizing binary to binary mapping

Corresponding Author :Aruna Tiwari, Computer Engineering Department,Shri G. S. Institute of Technology and Science (SGSITS), 23, Park Road, Indore 452003 (M.P.) INDIA

algorithm, convergence problem has been resolved as compared to backpropagation algorithm. Thus training time has been reduced. The use of integer weights and thresholds reduces prediction time also, as computations have been reduced.

In section 2 we discuss the basic concepts for extending the FCLA framework. In section 3 and 4, we present the formulae used under training and training algorithm of FCLA. In section 5, the extension of the FCLA framework is presented. Section 6 gives one illustrative example and in section 7 performance comparison is given, In section 8 we give concluding remarks.

BASIC CONCEPTS

Let $s=\{x_1,x_2,\ldots,x_m\}$ are the training examples. The proposed learning algorithm learns the classification function f(x) that takes these training examples and classifies it into one of k-classes: $f(x) \in \{c_1, c_2,\ldots c_k\}$. To learn this classification function, the algorithm analyzes a set of training examples $\{(x_1,f(x_1)), (x_2,f(x_2)), \ldots , (x_m,f(x_m))\}$. Each training example is a pair consisting of a description of an object x_i and its correct classification, $f(x_i)$.

The FCLA algorithm is designed for solving any binary (2-class) classification problems in three layer network structure as shown in fig 1.

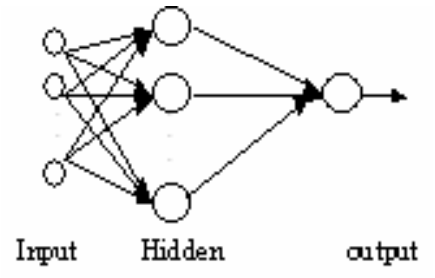

Fig. 1 : FCLA Three layer network structure

For each of the k classes, FCLA[2] algorithm can be applied separately for the training of hidden layer. Thus for each of the k-classes the FCLA algorithm can be applied in parallel in order to find out the hidden layer neurons with respect to each and every class. For combining the outputs of the hidden layer neurons, FCLA approach can be extended for the training of output layer by using either of the two coding schemes: OPC or ECOC and three layered network structure is formed as depicted in the figure 2.

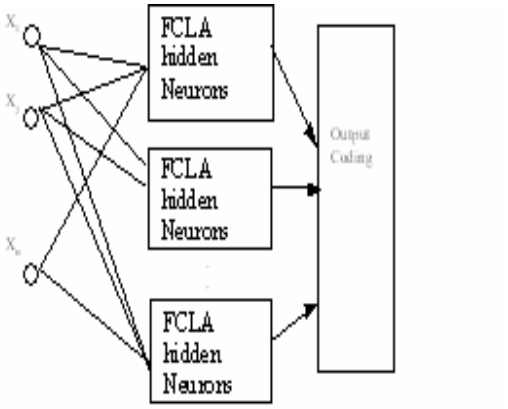

Fig. 2 : FCLA Three layer network structure used for multi-class problem

For deciding the output codes for each of the class, let $s_1,s_2,\ldots s_k$ be k distinct binary strings of length L. The length of the string will depend on the type of decomposition method used: OPC or ECOC. We call each string S_i the codeword for class c_i. Now define L hypotheses i.e. $f_1,f_2,\ldots,f_l$.

For OPC, $f_1,f_2,\ldots,f_k$ hypotheses are learned, one function f_i is defined for each class, such that $f_i(x)=1$ if $f(x)=c_i$ and zero otherwise. During learning, a set of hypotheses , $\{f_1,f_2,\ldots,f_k\}$ is learned. To classify a new example, x′, we compute the value of $f_i(x')$ for each i. The predicted value of f(x′) is the class c_i for which $f_i(x')$ is generating 1.

For ECOC, L hypotheses $f_1,f_2,\ldots,f_l$ for a class c_i if i=1, then $f_i=1$ for all i=1 to L otherwise there are alternating runs of 2^{k-i} zeroes and 2^{k-i} ones.

During learning, the hidden layer neurons are trained using two class learning algorithm to learn each of g_j function of $x_1,x_2,\ldots,x_m$ examples. The output layer neurons are trained depending on the coding scheme used for the classification OPC or ECOC, presented in the next section. The output layer have L hypotheses $\{f_1,f_2,\ldots,f_l\}$.

To classify a new example, x′, we apply each of the learned function g_j to compute binary string $s'=<f(x_1'), f(x_2'), \ldots , f(x_m')>$. Then we determine which codeword s_i is nearest to this s′ . The predicted value of f(x′) is the class c_i corresponding to the nearest codeword (having minimum Hamming distance) s_i.

FORMULAE USED: FAST COVERING LEARNING ALGORITHM

While constructing the BNN, suppose that $\{x_1, x_2,\ldots,x_v\}$ are v (true) vertices included in one hypersphere. The centre is defined as follows[2]:

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

three radii are defined as follows:

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

$$r_2^{\,2} = r_1^{\,2} + 1 \quad (3)$$

$$r_3^{\,2} = r_2^{\,2} + 1 \quad (4)$$

formulae for weights and threshold value of a neuron:

$$w_j = 2\sum_{k=1}^{v} x_i^k - v \quad (5)$$

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

$$t_2 = t_1 - v \quad (7)$$

$$t_3 = t_1 - 2v \quad (8)$$

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 C_k, take set of true vertices $(x_1,x_2\ldots x_m)$, each vertex is n-bit long represented as x_i^j, where $1 \le j \le n$.
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 (x_i) therefore evaluate following parameters-
-Center C (using equation (1))
-Radius r_1, r_2 ,r_3 (using equations (2), (3), (4))
-Weights (w_1 , w_2 ,.... w_n) represented as weight vector W (using equations (5))
-Thresholds(t_1,t_2,t_3) (using equations(6), (7), (8))
else
begin
-check this input data(x_i) with respect to the existing neurons
-for each of the p^{th} neuron do the following checks

<Cond1> if($Wx_i >= t_1$) then
-this input is already covered by the p^{th} neuron so simply exit & take next input(match region)

<Cond2> if($t_2 <= Wx_i <= t_1$)
-input data is within the claim region
-update the parameters of p^{th} neuron by using the formulae in section 3
-center C, radius, weights, threshold
-exit & take next input

<Cond3> if($t_3 > Wx_i$)
-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($t_3 <= Wx_i < t_2$)
-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.

B. Training the output layer

According to FCLA[2], at the output layer a single neuron is needed to collect the outputs of all the hidden neurons with respect to a two class problem as depicted

in fig.1. Let w_j^o represents the weights from j[th] hidden neuron to the o[th] output neuron. The total number of neurons for a given class are 'nc', out of which q represents the number of hidden neurons learned true vertices with generalization and the remaining (q+1,…nc) are the neurons which learned the false vertices. The weights and threshold of the output neurons are assigned as follows:

$$w_j^o = \begin{cases} 1 & if\ j=1,\ldots,q \\ -q & if\ j=q+1,\ldots,nc \end{cases}$$

and threshold of the neuron can be assigned as

$$t^o = 1 \qquad (9)$$

EXTENSION OF FCLA FRAMEWORK

We now use coding schemes for extending the FCLA framework for solving classification problems figure 3. We use two coding schemes for the construction of output layer : (1) OPC scheme, (2) ECOC scheme. The number of neurons required at the output layer depends on the coding scheme used.

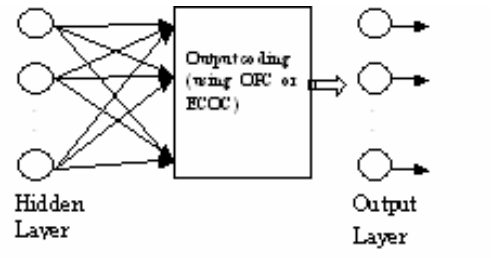

Fig. 3: Partial network showing the use of coding schemes for training the output layer

A. Construction of hidden layer

For a given K-Class problem $\{G_1,G_2,\ldots\ldots\ldots G_k\}$, for each & every class, we separately apply FCL[2] Algorithm 1. Thus hidden neurons are evaluated for each of the classes. After this, for collecting the outputs of the hidden neurons, we propose the approach in the next section.

B. Training Of Output Layer

The outputs generated by the hidden layer are combined at the output layer. The number of Output neurons are decided on the basis of the coding scheme used OPC or ECOC. As stated earlier, in OPC , the number of neurons are equal to the number of classes i.e. K. In ECOC, the number of neurons are 2^{k-1}-1. Thresholds of the output neurons are set to 1 in both the schemes. Further weight setting is done as follows:

1. OPC: Weight values for the i[th] class from j[th] neuron of hidden layer to the q[th] neuron of output layer is decided as follows:

$w_{i_{jq}} = 1$ if i=q;

$= 0$ otherwise

2. ECOC: Weight setting is done using following algorithm:

Algorithm 2

1. For each of the i[th] class
2. For each of the j[th] hidden layer neuron with respect to this class
3. Make the following assignment :
   current_op_neuron=1
4. For each of the q[th] output layer neuron
5. For the current_op_neuron to the
   (current_op_neuron+2^{k-i}-1)
   Assign weight value: $w_{i_{jq}} = 0$
6. For subsequent output neuron to the
   (current_op_neuron+2^{k-i}-1)
   Assign weight value: $w_{i_{jq}} = 1$
7. Repeat the steps 5 to 6 for each of the output neuron.
8. Repeat the steps 3 to 7 for each of the hidden neuron.
9. Repeat the steps 2 to 8 for each of the class.

ILLUSTRATIVE EXAMPLE

We illustrate the proposed approach with an example mentioned below:
Approximation of the following regions mentioned as A, B, C, D, E in the figure can be done by 6*6 grid. Table 1 gives the approximation of these regions through 6-bit binary values.

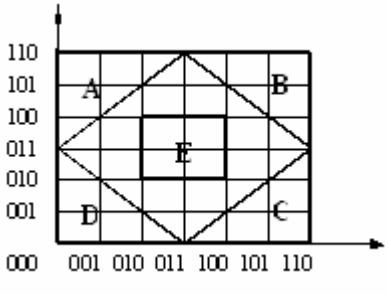

Fig. 4: Approximation of regions

Table 1:Data sets with respect to the approximated regions.

Intput datas	Region/Classes
{000100, 000101, 001101, 000011, 001100, 010101}	A
{100101, 101101, 101100, 011101, 100100, 101011}	B
{100000,101000, 101001, 011000, 100001, 101010}	C
{000000, 001000, 000001, 000010, 001001, 010000}	D
{010010, 010011, 011010, 011011}	E

Applying Algorithm 1 of section 2, the results of the construction of hidden layer is as follows:

Table 2 : Hidden layer solution

Inputs	Neurons	W_1	W_2	W_3	W_4	W_5	W_6	t_1	t_2	t_3	Region/ classes
{000100,	1	-5	-3	-1	5	-5	1	3	-2	-7	
000101,001101,000011,	2	-1	-1	-1	-1	1	1	2	1	0	A
001100, 010101}											
{100101,101101,	1	4	-4	0	4	-4	0	8	4	0	
101100,011101, 100100,	2	-1	1	1	1	-1	1	4	3	2	B
101011}	3	1	-1	1	-1	1	1	4	3	2	
{100000,101000,	1	5	-5	1	-5	-3	-1	3	-2	-7	
101001,011000, 100001,	2	-1	1	1	-1	-1	-1	2	1	0	C
101010}											
{000000,001000,											
000001,000010, 001001,	1	-6	-4	-2	-6	-4	-2	-4	-10	-16	D
010000}											
{010010,010011,011010,01101	1	-4	4	0	-4	4	0	8	4	0	E
1}											

Output layer weights for two methods:

Table 3 : Ouput layer weights and thresholds using OPC (One Per Class).

Hidden layer Neuron/output layer neurons	f_1	f_2	f_3	f_4	f_5	Thresholds	Regions/ classes
1	1	0	0	0	0	1	
2	1	0	0	0	0	1	A
1	0	1	0	0	0	1	
2	0	1	0	0	0	1	B
3	0	1	0	0	0	1	
1	0	0	1	0	0	1	
2	0	0	1	0	0	1	C
1	0	0	0	1	0	1	D
1	0	0	0	0	1	1	E

Table 4: output layer weights using ECOC (Error Correcting Output Code).

Hidden layer Neuron/output layer neuron	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	regions OR classes
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	A
1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	
2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	B
3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	
1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	C
2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	D
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	E

Next, tables 3 and 4 are depicted through the figures. As discussed in section 2, figure 2, three layered network structure is formed : input layer, hidden layer and output layer. Input layer doesn't contain any processing element, these are just nodes for providing inputs to the hidden layer. Hidden and output layers contains the neurons. With respect to table 3, network structure formed is depicted in figure 4. Network structure for Table 4 is shown in figure 5.

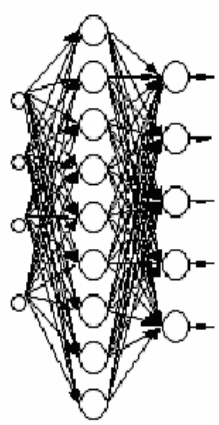

Fig. 4: Example Solution using OPC scheme

We make use of the Fisher's Iris data set for comparing the performance of the coding schemes used OPC and ECOC for the designing of classifiers in FCLA. Fisher's Iris Data Set contains 150 patterns for representing three classes[10]. There are 50 patterns of each class. There are four properties on the basis of combination of these properties, the classification have been done. For applying the inputs to the network the each of the four properties of the original pattern have been represented by 7-bit binary equivalent. Thus the

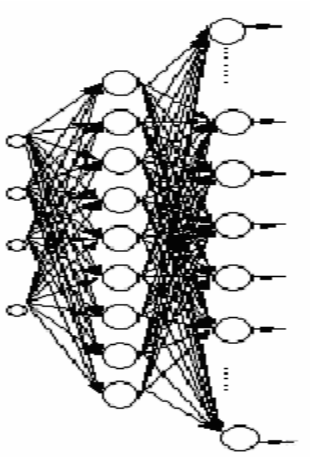

Fig. 5: Example Solution using ECOC scheme

PERFORMANCE COMPARISION

input contains total of 28-bits. Hidden layer neurons have been found out by using FCLA approach. Total of 32 neurons are required in the hidden layer. For Setosa : 17 neurons are needed. For Versicolor: 9 neurons and for Virginica: 6 neurons are needed. The number of output neurons are 3 for both the coding schemes used OPC or ECOC. The weights and thresholds of the output layer neurons are given in the tables 5 and 6 as follows :

Table 5 : Output layer neurons when using OPC scheme

Classes/neurons	f_1	f_2	f_3	Threshold
(1) Setosa	1	0	0	1
(2)Versicolor	0	1	0	1
(3) Virginica	0	0	1	1

Table 6 : Output layer neurons when using ECOC scheme

Classes/neurons	f_1	f_2	f_3	Threshold
(1) Setosa	1	1	1	1
(2) Versicolor	0	0	1	1
(3) Virginica	0	1	0	1

For testing over these pattern, we split each of the 50 patterns for each of the class 40/10 (train/test) data. Testing results show that ECOC performs better in terms of classification accuracy. For Setosa and Versicolor , ECOC is gives 100% accuracy(i.e. classifying all the 10 samples properly). For Virginica, 80% accuracy is achieved with ECOC. Using OPC with the same case, results are not satisfactory.

CONCLUSION

In this paper, we extend FCLA[2] method for multi-class problems by designing classifiers using coding schemes. The hidden layer trained is in modular form. Thus modules in the hidden layer corresponding to each class can be trained independently[4] in parallel, thus reduces training time. For output layer training, the paper has examined the use of Error correcting coding and One Per Class coding scheme for binary to binary mapping learning algorithm. The performance of the method has been compared on the Fisher's well-known Iris dataset. The results shows that ECOC gives more classification accuracy as compared to OPC.

REFERENCES

1. Thomas G. Dietterich, Ghulum. Bakiri,1995. Solving Multiclass Learning Problems via Error-Correcting Output Codes : Journal of Artificial Intelligence Research, Vol. 2 : 263-286.
2. Di Wang and Narendra S. Chaudhari, 2004. An Approach for Construction of Boolean Neural Networks Based on Geometrical Expansion : Neurocomputing, vol. 57, pp :455-461.
3. 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.
4. Rangachari Anand, Kishan Mehrotra, Chilukuri K. Mohan and Sanjay Ranka, 1995. Efficient Classification of multiclass problem using Modular Neural Network : IEEE transactions on Neural Networks, vol.6, pp : 117-124.
5. Francesco Masulli., Giorgio Valentini, 2000. Comparing Decomposition Methods for Classification : Proc. Of International Conference on Knowledge-based Intelligent Engineering Systems & Allied Technologies, Vol. 2 : 788-791.
6. Erin L. Allwein, Robert E. Schapire, Yoram Singer, 2000. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers : Proc. Of International Conference on Machine Learning , pp : 9-16.
7. Francesco Masulli, Giorgio Valentini, 2000. Effectiveness of error-correcting output codes in multiclass learning problems : In Proc. Of MCS (2000), First International Workshop on Multiple Classifier Systems, Cagliari, Italy.
8. Atsushi Yamamoto, Toshimichi Saito, 1997. An improved Expand-and-Truncate Learning : Proc. Of IEEE International Conference on Neural Networks (ICNN), Vol. 2, pp : 1111-1116.
9. Koby Crammer, Yoram Singer, 2000. On the learnability and design of output codes for multiclass problems :In proceedings of Thirteenth Annual Conference on Computational Learning Theory, pp : 35-46.
10. Kishan Mehrotra, Chilukuri K. Mohan and Sanjay Ranka, 1997. Elements of Artificial Neural Networks : Cambridge, MA:MIT Press.