ARTIFICIAL NEURAL NETWORK-BINARY LABEL CLASSIFICATION

Saurav Singh[1], MahimaParashar[2]

[1]*Electronics and Communication Engineering, Maharaja Surajmal Institute of Technology, New Delhi*
[2]*Computer Science, Maharaja Surajmal Institute of Technology, New Delhi*

Abstract

Artificial Neural Networks (ANNs) are a family of models which try to mimic the network of neurons in the brain to process information. It is a system of interconnected neurons which exchange the messages between each other. When too many classifications are involved, the ANN tends to use up a lot of memory space for bigger weight matrices and more output nodes corresponding to more classification labels. The goal of our research is to analyze the existing ANN model and develop an algorithm that can give output in binary format instead of dedicating one output node to each label. This will help in reducing the number of output nodes and weight matrices and reduce the memory space required while maintaining the accuracy of the network.

Keywords: *Binary, Classification, Network, Neural, Supervised*

----------***----------

1. INTRODUCTION

Artificial Neural Networks are electronic networks of neurons based on the neural structure of the brain. ANNs are majorly used in supervised multi-class classification applications. Artificial neural networks are generally presented as systems of interconnected nodes which exchange messages between each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

In supervised learning, we have to teach the network how to respond by feeding the input samples and the corresponding outputs. Once the network aligns itself to the sample data, it can be used to classify the input data into the defined set labels. A Neural Network may contain an input layer, one or more hidden layers and an output layer. The connections between these layers have numerical weights which are aligned with the sample data at the time training. Figure 1 illustrates the general neural network model.

An input data can be classified by using forward propagation algorithm. In forward propagation, the output of each layer is calculated by the sum of products of weights and inputs of each node of the previous layer through an activation function. A common choice of activation function for multi-layer neural network is logistic function or sigmoid function.

$$\varphi(z) = \frac{1}{1 + e^{-z}}$$

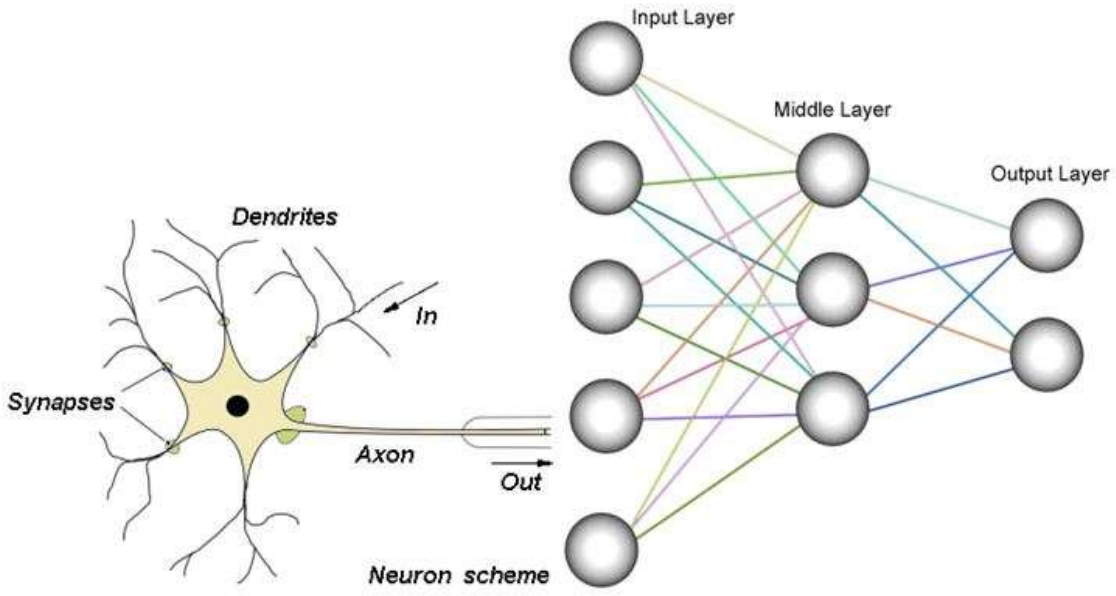


Fig. 1 Neural Network Model

2. FORWARD PROPAGATION

The forward propagation can be described by the following equations:

$$Z_n^{(l)} = \sum_{k=0}^{k} \Theta_{kn}^{(l-1)} a_k^{(l-1)}$$

$$a_n^{(l)} = \varphi(z_n^{(l)})$$

where, $a_o^{(l)} = 1$

$$a_k^{(1)} = x$$
$$y_k = a_k^{(L)}$$

3. BACKWARD PROPAGATION

For training the network, back propagation algorithm is used. [1] [2] Since back propagation uses the gradient descent meth-od, one needs to calculate the derivative of the squared error function with respect to the weights of the network. Assuming one output neuron, the squared error function is:

$$E = \frac{1}{2}(t - y)^2,$$

where,
E is the squared error,
t is the target output for a training sample, and
y is the actual output of the output neuron.

The factor of 1⁄2 is included to cancel the exponent when differentiating. Later, the expression will be multiplied with an arbitrary learning rate, so that it doesn't matter if a constant coefficient is introduced now.

For each neuron j, its output O_j is defined as.

$$O_j = \varphi(net_j) = \varphi\left(\sum_{k=0}^{n} w_{kj} O_k\right)$$

The input net_{ij} to a neuron is the weighted sum of output Ok of previous neurons. If the neuron is in the first layer after the input layer, the O_k of the input layer are simply the inputs x_k to the network. The number of input units to the neuron is n. The variable w_{ij} denotes the weight between neurons i and j.

The activation function φ is in general non-linear and differentiable. A commonly used activation function is the logistic function:

$$\varphi(z) = \frac{1}{1 + e^{-z}}$$

which has a nice derivative of:

$$\frac{d\varphi}{dz}(z) = \varphi(z)(1 - \varphi(z))$$

3.1 Finding the Derivative of the Error

Calculating the partial derivative of the error with respect to a weight w_{ij} is done using the chain rule twice:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial O_j} \frac{\partial O_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ij}}$$

In the last term of the right-hand side of the following, only one term in the sum net_j depends on w_{ij}, so that,

$$\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}\left(\sum_{k=1}^{n} w_{kj} O_k\right) = O_i$$

If the neuron is in the first layer after the input layer, O_i is just x_i.
The derivative of the output of neuron j with respect to its input is simply the partial derivative of the activation function (assuming here that the logistic function is used):

$$\frac{\partial O_j}{\partial net_j} = \frac{\partial}{\partial net_j}\varphi(net_j) = \varphi(net_j)(1 - \varphi(net_j))$$

This is the reason why back propagation requires the activation function to be differentiable.

The first term is straightforward to evaluate if the neuron is in the output layer, because then $O_j = y$ and,

$$\frac{\partial E}{\partial O_j} = \frac{\partial E}{\partial y} = \frac{\partial}{\partial y}\frac{1}{2}(t - y)^2 = (y - t)$$

However, if j is in an arbitrary inner layer of the network, finding the derivative E with respect to Oj is less obvious.

Considering E as a function of the inputs of all neurons L= u,v…..,w receiving input from neuron j,

$$\frac{\partial E(O_j)}{\partial O_j} = \frac{\partial E(net_u, net_u, \ldots\ldots, net_u,)}{\partial O_j}$$

And taking the total derivative with respect to Oj, a recursive expression for the derivative is obtained:

$$\frac{\partial E}{\partial O_j} = \sum_{l \in L}\left(\frac{\partial E}{\partial net_l}\frac{\partial net_l}{\partial O_j}\right) = \sum_{l \in L}\left(\frac{\partial E}{\partial O_l}\frac{\partial O_l}{\partial net_j} w_{jl}\right)$$

Therefore, the derivative with respect to O_j can be calculated if all the derivatives with respect to the outputs O_l of the next layer – the one closer to the output neuron – are known.

Putting it all together:

$$\frac{\partial E}{\partial w_{ij}} = \delta_j O_i$$

With,

$$\delta_j = \frac{\partial E}{\partial O_j}\frac{\partial O_j}{\partial net_j}$$

$$= \begin{cases} (O_j - t_j)O_j(1 - O_j) & if\ j\ is\ an\ output\ neuron \\ \left(\sum_{l \in L} \delta_l w_{jl}\right) O_j(1 - O_j) & if\ j\ is\ an\ inner\ neuron \end{cases}$$

To update the weight w_{ij} using gradient descent, one must choose a learning rate, α. The change in weight, which is added to the old weight, is equal to the product of the learning rate and the gradient, multiplied by -1:

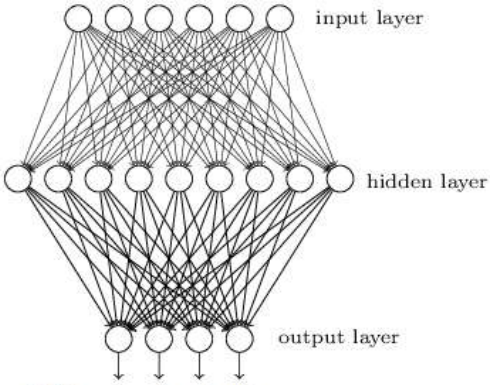
$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}}$$

After the ANN has been trained, forward propagation can be used to determine the class of any input data. For multi-class classification, each node is trained to indicate one label using one-vs-all methodology. The output of activation function of each node in the output layer signifies how much confident the ANN is that the input data belongs to that particular label. Thus the input data can be classified into the label with the highest level of confidence.

Using this method can give us ANNs with very high accuracy. But as the number of classes/labels increases, the number of nodes corresponding to each label in the output layer also increases. This leads to higher usage of memory space and increase the electronic network components. Also, the increase in output nodes increases the weight matrix and the computation cost. An alternate method is presented in this paper which reduces the output nodes for same number of labels, reduces the memory required and computation cost while maintaining the accuracy and gives the output in binary format so that the output can be directly used by other processors or DSP units directly.

4. PROPOSED

To reduce the output nodes, each label is represented by a binary pattern of output nodes. Once each label has been assigned a binary number, each output node of network can be trained accordingly. If there are L number of labels, m number of nodes are required to represent the output bits, where L ⩽ 2m, that is, ceil ($\log_2$ L) output nodes are required. An example of digit recognition has been implemented using the proposed method. In figure 2, the encoding of each label from 0 to 9 has been assigned a binary number and only 4 output nodes required to represent the 10 labels.



Labels	y1	y2	y3	y4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Fig 2. Proposed label encoding technique

Therefore, the label '0' is represented by y1y2y3y4 = '0000'; the label '1' is represented by y1y2y3y4 = '0001'; the label '2' is represented by y1y2y3y4 = '0010' and so on. During back propagation, the output matrix for training data for each output node (i.e., y1, y2, y3 and y4) is taken as shown in figure 2. Figure 3(a) and 3(b) shows the change in output matrix before and after the applying the proposed method.

In the conventional method, while running the forward propagation algorithm, the activation function of each output node gives the level of confidence that the ANN have for the input data to belong into that particular class and the node showing maximum confidence is chosen to be the winner. Previously, only one node among the L nodes with giving the output as 1 and rest of the L-1 nodes were giving the output 0. Since now the output of each node is a part of binary pattern of output nodes, it is possible that there will be more than one 1's for a particular label as shown in figure 2. Thus we cannot choose the output node with maximum confidence as 1 and rest of the nodes as 0's.

labels	y1	y2	y3	y4	y5	y6	y7	y8	y9	y10
0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0

(a)

labels	y1	y2	y3	y4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

(b)

Fig. 3: 3(a) Output matrix before applying the proposed method. 3(b) Output matrix after applying the proposed method

The most general, and most natural, framework in which to formulate solutions to pattern recognition problems is a statistical one, which recognizes the probabilistic nature both of the information we seek to process, and of the form in which we should express the results [4]. This problem is addressed by the concept of threshold. A confidence threshold is set for each output node. If the output shows confidence more than the threshold level, the output of that node is taken to be as 1; else the output is taken as 0. The optimum value for the threshold is 50% or 0.5 as observed experimentally. This concept of threshold helps us to decode the confidence level of output layer into a binary pattern indicating an output label.

5. EXPERIMENTAL STEPS

The experimental steps of the proposed method are as discussed below:

1. A three layer Artificial Neural Network is designed with one input layer, one output layer and one hidden layer for digit recognition system with randomly initialized weights.
2. The output labels are encoded into binary patterns as shown in figure 2.
3. Input data samples are used for forward propagation to calculate the system output with random weights to find out the output error (t1, t2, t3, t4) with respect to desired outputs (y1, y2, y3, y4). The input data consist of 20X20 resolution images of digits.
4. Back propagation algorithm is used to train the network to adjust the weights to give the desired output.
5. Finally, the system is ready to operate as a digit recognition system. A separate test set is used to calculate the accuracy of the proposed method. The accuracy has been calculated by taking percentage of correct predictions to total predictions,

$$accuracy = \frac{(correct\ predictions)}{(total\ predictions\ made)} \times 100$$

The same experiment with same training data and test data have been conducted once without a hidden layer and once with the conventional method for comparison.

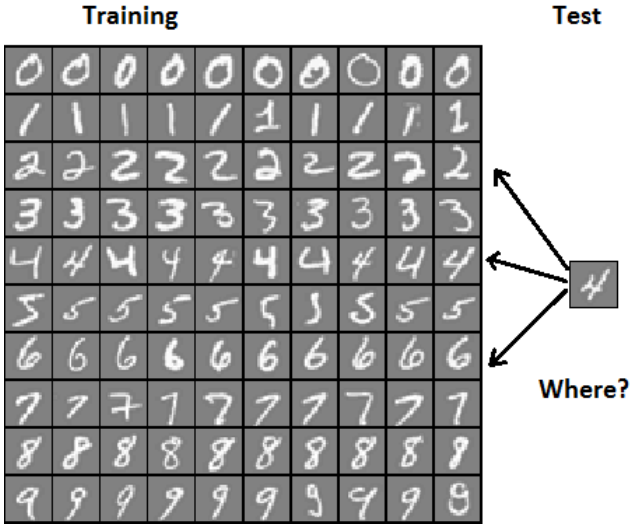


Fig 4. Comparing a new image with the database

6. MEMORY USAGE

Let the number of nodes in input layer, hidden layer and output layer in the conventional method be V, U and L respectively. The weight matrix has the size (U×L) .

As the proposed method aims to reduce the number of nodes in the output layer and the corresponding weight matrix, thus if there are L number of labels, at least m number of nodes are required to represent the output bits, where L $\leqslant$ 2m, that is, ceil (log2L) output nodes are required.

Thus, the reduction in the output nodes for L labels is given by,

$$D = L - ceil(log_2 L)$$

The size of the weight matrix of the proposed method is $(U \times ceil(log_2 L))$, thus the reduction in the weight matrix is given by,

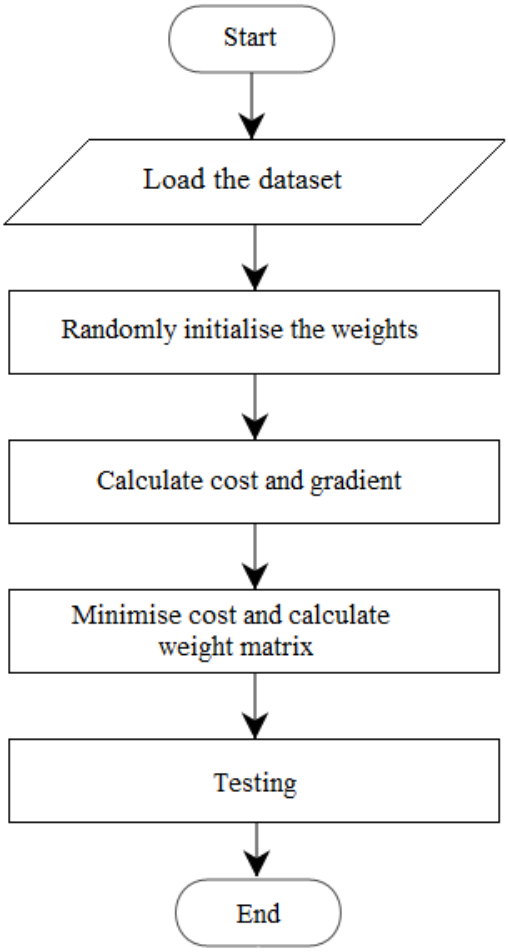
$$W = U(L - ceil(log_2 L))$$

i.e. $$W = U.D$$

Therefore, the total reduction in the memory usage is the addition of the reduction in memory reduction of the weight matrix and memory reduction in the output labels, i.e. (D+W),which is equal to (U.D+D),

i.e. $$D(U + 1)$$

7. FLOWCHART



8. EXPERIMENTAL DATA

In the database, 500 different images of each of 10 digits (total 5000 images) with variations in writing style, thickness and angle of tilt are considered. Each image is a 20X20 resolution grayscale image. A preview image of the Database of Digits is as shown in figure.

Fig 6 Database of hand written digits

9. RESULT

Computational Efficiency for different types of models is shown in Table 1.

Table 1: Computational Efficiency Of Different Models

Network Model	Number of output nodes	Type of model	Accuracy (in %)
Logistic Regression- One Vs all with one Input layer and one Output layer	10	Linear	94.94
Logistic Regression- One Vs all modified by the proposed encoding technique with one Input Layer and one Output layer	4	Linear	74.52
Conventional Artificial Neural Network with one Input layer, one Hidden layer and one Output layer	10	Non-Linear	97.52
Artificial Neural Network modified by the proposed encoding technique with one Input layer, one Hidden layer and one Output layer	4	Non-Linear	96.58

10. CONCLUSION

The presented method can be used to design any feed forward Artificial Neural Network with reduced output nodes while maintaining high accuracy. Moreover, the output of the ANN is in binary format that can be directly fed into processors, DSP units, microcontrollers and other digital systems for further processing. The electronic network is reduced in size and memory space required to store L output label status and weight matrix has also reduced. The high accuracy of the ANN for digit recognition (which comes out to be 96.58% for the experimental dataset) is proving the efficacy of the pro-posed approach.

It is observed that the accuracy of a logistic regression model has drastically decreased after it has been modified by the proposed encoding technique. This is because the encoding technique demands a non-linear model and logistic regression offers a linear network model.

REFERENCES

[1] Raúl Rojas, "The back propagation algorithm of Neural Networks," in *A Systematic Introduction*

[2] Simon Haykin, "Multilayer Perceptron" in N*eural Networks and Machine Learning*, 3rd ed.

[3] Michael A. Nielsen, *Neural Network and Deep Learning*, Determination Press, 2015

[4] Christopher M. Bishop, "Statistical Pattern Recognition" in *Neural Networks for Pattern Recognition*, Claredon Press, 1995

[5] Lecun Y., Boser B., Denker, J., Henderson D., Howard R., Hubbard W., &Jackel L. (1990). "Handwritten digit recognition with a backpropagation network". Advances in Neural Information Processing Systems 2, editor Touretzky D., pp.396-404. San Mateo, CA: Morgan Kaufmann.

[6] Lecun Y., Jackel L.D., Bottou L., Cortes C., Denker J.S., Drucker H., Guyon I., Muller U.A., Sackinger E., Simard P., Vapnik V. (1995). "Learning algorithms for classification: a comparison on handwritten digit recognition". Neural Networks: The Statistical Mechanics Perspectives, editors Oh J.H., Kwon C., & Cho S., pp.261-276. World Scientific.

[7] Lecun Y., Bottou L., Bengio Y., Haffner P. (1998). "Gradient-Based Learning Applied to Document Recognition". Proceedings of the IEEE, 86(11), pp.2278-2324.