RESEARCH ARTICLE OPEN ACCESS

Review of Artificial Neural Network Technique for Restoring Image

Sumit Bhardwaj[1], Arun Kumar[2], Dr. Vinod Shokeen[3]
Electronics and Communication Engineering[1, 2&3]
Amity University Uttar Pradesh (AUUP) [1&2]
Dehradun Institute of Technology[3]
Greater Noida - India

ABSTRACT
Artificial neural networks are applied many areas successfully because of their ability to learn, ease of implementation and fast real-time operation. ANNs are inspired by the biological tool which learns, retains and uses the subsequent processing. In this paper we will study how ANN can be implemented and about the algorithm used by ANN for a network along with the mathematical calculations involved.

Keywords:- *ANN, MLP, BP, LM, GDBP, GDMBP, GDMALRBP*

I. INTRODUCTION

ANN is a mathematical or computational model that attempts to simulate the functional features of biological neural network [1]. ANNs have been applied to de-noise or to restore defected or degraded images for many applications. It is an adaptive system changing its structure during a learning phase. They are composed of simple elements inspired by biological nervous systems working in parallel. As in nature, the network function is determined largely by the connections between elements.

II. ARTIFICIAL NEURAL NETWORK

Most of the ANNs are trained o adjusted for a particular input, leading to a specific target output for a desired application. Selectively many such input/target pairs are used which is done in such a supervised learning method to train a specified network. For example, Batch training of a network proceeds by constructing weight and bias changes based on an entire set (batch) of input vectors for application particularly. As a result incremental training changes the weights and biases of a network as needed after presentation of each individual input vector for the same and this incremental training is sometimes referred to as on line or adaptive training of network. For this batch learning, the error is calculated for multiple training samples before the weights in the network are updated and this is calculated for the entire dataset before back propagation is performed in batch learning. This means the average or 'true' error gradient of the dataset is used to update the learning parameters. The 'true' error gradient refers to the fact that the error of the entire dataset is used rather than the error of a single sample of the dataset.

However Modern ANNs are non-linear statistical data modelling tools, which are usually used to model complex relationships between inputs and outputs or to find patterns in data.

III. MULTILAYER PERCEPTRON (MLP)

All The smallest individual component of an ANN is the perceptron which has the ability to learn, commonly referred as node, considered to be a basic binary classifier. A single perceptron is only capable of discriminating an input feature vector x between two linearly separable classes 0 and 1. Multilayer perceptrons are composed of a network of connected perceptrons. As shown in Figure 1, it is represented as a directed graph with one or more inputs and outputs. The typical structure is organized into multiple layers of nodes. Each and every node in a layer has in incoming connection from all nodes in its preceding layer for a architecture completely, but nodes within the same layer are not connected with one another. Thus the output of a single node in one layer is a function of all connected nodes in the preceding one and those which are not at the input or output of the network are referred to as hidden [2]. This process is called a feed forward as the inputs are fed forward from the input of the perceptron to its output.

IV. ACTIVATION FUNCTION

The activation function is also known as the basis or squashing function. In most cases, this is a differentiable

function which adds a non-linear component to the MLP architecture. In absence of this non-linearity, the network would be reducible to a single layer of perceptrons. The activation function was originally chosen to be a relay function, but for calculation a hyperbolic tangent (tanh) or a sigmoid function are mostly used, and is given as

$$\tanh(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \quad (1)$$

The ANNs are made of arrangements of processing elements (neurons). The artificial neuron model is given by

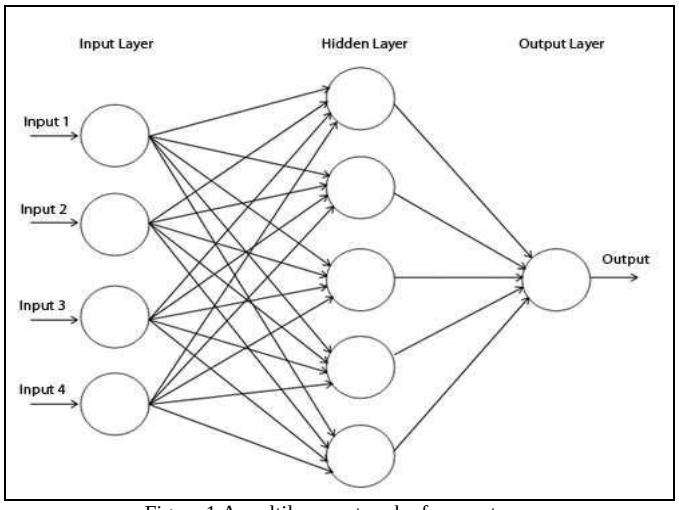

Figure.1 A multilayer network of perceptrons

$$y_k = \Phi\left(\sum_{j=1}^{n} w_{kj}x_j + b_k\right) \quad (2)$$

Where wkj are the connection weights, bk is known as threshold parameter, xj is the input vector and yk is the output of the kth neuron, the Φ is the function that provides the activation for the neuron. ANN solves nonlinear problems, if we use nonlinear activation functions for the solving the hidden and/or the output layers [3].

The MLP network architecture involves fixing of hidden layers numbers and neurons (nodes) for all the layers present. Also the activation functions for each layer are chosen at this stage, and thus are considered to be known before. All weights and biases are the unknown parameters to be estimated. The most well-known are Back-Propagation (BP) of which there are several version including Levenberg-Marquardt (LM) algorithms.

V. BACK-PROPAGATION (BP) ALGORITHM

Back-propagation (BP) is the most common method of training a neural network. Generalizing the least mean square algorithm used in adaptive filtering, BP utilizes the optimization method known as gradient descent to adjust the weights of each connection in the network to minimize the average error [4]. In the simplest form, error is defined as the difference between the desired output dj of the network and its actual output yj when presented with a training example n. Thus, for a single output neuron j, the error ej is defined as

$$e_j(n) = d_j(n) - y_j(n) \quad (3)$$

BP is focused on to minimize ej within the training set of size N, the average error Eav.

$$E_{av} = \frac{1}{N}\sum_{n=1}^{N} E(n) \quad (4)$$

At the output layer this error is easily measured. There are only two passes present in Error Propagation, known as forward pass and a backward pass, respectively. In general for forward pass the input vector is applied to the sensory nodes of the network and its effect propagates through the network layer by layer and a set of outputs is produced as the actual response of the network finally. While during forward pass the synaptic weight of the networks are all fixed every time, whereas during the back pass the synaptic weights are all adjusted in accordance with an error-correction rule generally. Therefore the actual response of the network is subtracted from the desired response to produce an error signal for this. This error signal is then propagated backward through the network against the direction of synaptic conditions and the synaptic weights are adjusted to make the actual response of the network move closer to the desired response of the network [4].

The BP algorithm follows the delta rule where the delta error in the output layer is trans-formed by the derivative of the transfer function and is then used in the previous neural layer to adjust input connection weights.

A. *TRAINING*

Iteratively presenting it with examples of the correct known answers is a part of training a neuron. The objective of training is to find the set of weights between the neurons that determine the global minimum of error function. Involving decision regarding the number of iteration, selection of learning rate (a constant of proportionality which determines the size of the weight adjustments made at each iteration) and momentum values (how past weight changes affect current weight changes) [5]. There are two approaches to training: supervised and unsupervised. For supervised training, network processes the inputs and compares its resulting outputs against the desired outputs propagating the errors back through the

system causing the system to adjust the weights which control the network once both the inputs and the outputs are provided. So this process takes place again and again and the weights are continually updated in this process. The set of data which enables the training is called the, training set. Training sets need to be fairly large to contain all the needed information if the network is to learn the features that are important. During the training of a network the same set of data is processed many times as the connection weights are ever refined [6]. In unsupervised training, the system itself must then decide what features it will use to group the input data as the network is provided with inputs but not with desired outputs, referred to as self-organization or adaption of a network.

Here, we briefly describe the steps related to the BP algorithm. A MLP is formed by multiple layers of perceptrons which are the basic processing units of an ANN.

A simple perceptron is a single neuron trained by the perceptron algorithm is given as

$$O_x = g([w][x] + b) \quad (5)$$

Where [x] is the input vector, [w] is the associated weight vector, b is the bias value and g(x) is the activation function.

Such a setup, namely the perceptron will be able to classify only linearly separable data. Whereas in MLP, consists of several layers of neurons. Expression for output in a MLP with one hidden layer is given as:

$$O_x = \sum_{i=1}^{N} \beta_i g([w]_i[x] + b_i) \quad (6)$$

Where β_i is the bias value, wi weight value between the ith hidden neuron. The process of adjusting the weights and biases of a perceptron or MLP is called training. Also the perceptron algorithm for training simple perceptrons consists of comparing the output of the perceptron with an associated target value. The most common training algorithm used for MLPs is error BP.

The MLP is trained using (error) BP depending upon which the connecting weights between the layers are updated & this adaptive updating of the MLP is continued till the performance goal is met. MLP training is done in two passes- one a forward pass and the other a backward calculation with error determination and connecting weight updating in between. This method is adopted as it accelerates the speed of training and the rate of convergence of the MSE to the desired value of the algorithm. The steps are as below:

B. Initialization

First, we take weight matrix W with random values between [-1; 1] if a tan-sigmoid function is used as an activation function and between [0; 1] if log-sigmoid function is used as activation function. W is a matrix of CxP where P is the length of the feature vector used for each of the C classes.

Presentation of training samples: Let the input be pm = [pm1; pm2:::pmL]. The desired output is dm = [dm1; dm2:::dmL] giving the values of the hidden nodes are

$$net^h_{mj} = \sum_{i=1}^{L} w^h_{ji} p^{mi} + \Phi^h_j \quad (7)$$

We obtain the output from the hidden layer as:

$$o^h_{mj} = f^h_j(net^h_{mj}) \quad (8)$$

Here $f(x) = \frac{1}{e^x}$ or $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ depending upon the choice of activation function. The values of the output node can be obtained as:

$$O^o_{mk} = f^o_k(net^h_{mj}) \quad (9)$$

Forward Computation

For this the errors computation is done as

$$e_{jn} = d_{jn} - o_{jn} \quad (10)$$

The MSE is calculated as:

$$MSE = \frac{\sum_{j=1}^{M} \sum_{n=1}^{L} e^2_{jn}}{2M} \quad (11)$$

Error terms for the output layer are:

$$\alpha^o_{mk} = o^o_{mk}(1 - o^o_{mk}) e_{mn} \quad (12)$$

Error terms of the hidden layer are:

$$\alpha^h_{mk} = o^o_{mk}(1 - o^o_{mk}) \sum \alpha^o_{mj} w^o_{mj} \quad (13)$$

C. Weight Update:

Between the output and hidden layers

$$w^h_{kj}(t+1) = w^h_{kj}(t) + \mu \alpha^h_{mj} p_i + (w^o_{ji}(t+1) - w_{ji}) \quad (14)$$

Where μ is the learning rate.

A few of the methods used for MLP training includes:

1. Gradient Descent (GDBP)

In this BP method the training will continue as long as the network has its weight, net input, and transfer functions

present. Thus BP is used to calculate derivatives of performance with respect to the weight and bias variables. Each variable is adjusted according to gradient descent. The training will be stop if the maximum number of epochs (repetitions) is reached, the maximum amount of time has been exceeded, or the performance has been minimized to the goal.

2. *GRADIENT DESCENT WITH MOMENTUM BP (GDMBP):*

In this method BP is used to calculate derivatives of performance with respect to the weight and bias variables present. Getting every variable adjusted according to gradient descent with momentum a specific value and it depends on the previously changed weight or bias passing with every epochs with a given learning rate.

Gradient Descent with Momentum and Adaptive Learning

3. *RATE BP (GDMALRBP):*

This method is used to train any network as long as it constitutes weight, gross input and transfer functions have derivative functions associated with it. Each variable is adjusted according to gradient descent with momentum constant, and also depends upon previously changed weights of the algorithm. Decreasing the performance for every epoch towards the goal, then the learning rate is increased by a specific factor [7].

CONCLUSION

The primary objective of using the ANN is its ability to learn from the surrounding, retain it and use for subsequent processing. It is observed that the ANN based approach provides 6 to 26% improvement for a range of images considered compared to the other methods of restoration. The ANN is restricted by need of greater computational complexity, empirical nature of model development, and the affection to over-fitting.

REFRENCES

[1] Y. A. Al-Sbou, "Artificial Neural Networks evaluation as an image denoising tool", World Applied Sciences Journal 17 (2), pp. 218-227, 2012.

[2] D.Wang,T.Dillon and E.Chang, "Pattern learning based image restoration using Neural Networks", IEEE, 2002.

[3] A.Castro and D.S.Silva, "Neural Network based multiscale image restoration approach" SPIE-IS & T, vol.6497.

[4] C.Khare and K.Kumar, "Image restoration technique with Non Linear Filter", International Journal of Advanced Science and Technology, vol.39, pp. 67-74, 2012.

[5] K.Kumar and G.S.M. Thakur, "Advanced applications of Neural Networks and artificial intelligence: a review", I.J. Information Technology and Computer Science, pp.57-68, 2012.

[6] F.Gunther and S.Fritsch, "neuralnet: Training of Neural Networks", The R Journal, vol. 2, pp. 30-38, 2010.

[7] S.Haykin, "Fundamentals of Artificial Neural Network", 3rd edition, Pearson, New Delhi, 2003.