Formulation for Second Derivative of Generalized Delta learning Rule in Feed Forward Neural Networks for Good Generalization

[1] Shivpuje Nitin Parappa
Shri Venkateshwara University,
Gajraula , Amroha(Uttar Pradesh),
India.
(e-mail: shivpujenitin@gmail.com)

2 DR. Manu Pratap Singh
Institute of Computer & Information Science
Dr. B.R.Ambedkar University, Agra-282002
Uttar Pradesh, India
(e-mail: manu_p_singh@hotmail.com)

Abstract— Generalized delta learning rule is very often used in multilayer feed forward neural networks for accomplish the task of pattern mapping. A backpropagation learning network is expected to generalize from the training set data, so that the network can be used to determine the output for a new test input. This network uses the gradient decent technique to train the network for generalization. It evolves the iterative procedure for minimization of an error function, with adjustments to the weights being made in a sequence of steps. The first derivate of the error with respect to the weights identifies the local error surface in decent directions. Therefore for every different presented pattern, the network exhibits the different local error and the weights modify in order to minimize the current local error. In this paper, we are providing the generalized mathematical formulation for the second derivative of the error function for the arbitrary feed forward neural network topology. The new global error point can evaluate with the help of current global error and the current minimized local error. The weight modification process accomplishes two times, one for the present local error and second time for the current global error. The proposed method indicates that the weights, these are determines from the minimization of global error are more optimal with respect to the conventional gradient decent approaches.

Index Terms — Generalization, Pattern mapping networks, Back propagation learning network, decent gradient, and Conjugate descent.

1 Introduction

Generalization is different from interpolation, since in generalization the network is expected to model the unknown system or function from which the training set data has been obtained. The problem of determination of weights from the training set data is known as the loading problem [1]. The generalization performance depends on the size and complexity of the training set, besides the architecture of the network and the complexity of the problem [2]. Therefore, if the performance for the test data is as good as for the training data, then the network is said to have generalized from the training data. The feed forward neural network architecture is more commonly used to perform the task of generalization with pattern mapping network. In this pattern mapping network for generalization the algorithm for modifying the weights between the different interconnected layers is usually known as back propagation learning technique [3]. This algorithm is a supervised learning method for multi layered feed forward neural networks. It is essentially a gradient descent local optimization technique which involves backward error correction of network weights. It evaluates the derivatives of the error function with respect to weight in weight space for any given presented input pattern from the given training set [4]. It involves the iterative procedure for minimization of an error function, with adjustment to the weights being made in a sequence of steps. In each steps we can distinguish between two distinct stages. In the first stage, the derivatives of the error function with respect to the weights must be evaluated. In the second stage, the derivatives are then used to compute the adjustment to be made to the weights [10]. The first stage process, namely the propagation of errors backward through the network in order to evaluate derivatives, can be applied to many other kinds of network and not just the multi layer perceptron. It can also be applied to the error functions other than just the simple sum-of-squares, and the evaluation of other derivatives such as the Jacobin and Hessian metrics [11] and also, the second stage of weight adjustment with the calculated derivatives can be tackled using a verity of optimization schemes.

Despite the general success of back-propagation method in the learning process, several major deficiencies are still needed to be solved like convergence guarantee and convergence rate, nature of error, ill posing and over training. The convergence rate of back-propagation is very low and hence it becomes unsuitable for large problems. Furthermore, the convergence behavior of the back-propagation algorithm depends on the choice of initial values of connection weights and other parameters used in the algorithm such as the learning rate and momentum term. There are various other enhancements and modifications were also presented by different researchers [5-7] for improving the training efficiency of neural network based algorithms by incorporating the selection of dynamic learning rate and momentum [8-9]. The improvement in performance of back propagation is further consider with the evaluation of the Jacobin matrix [12], whose elements are given by the derivatives of the network outputs with respect to the inputs. The Jacobin matrix provides a measure of the local sensitivity of the outputs to change in each of the input variables. In general, the network mapping represented by a trained neural network will be non-linear, and so the elements of the Jacobin matrix will not be constant but depends on the particular input vector used.

The back propagated error is further used to evaluate the second derivatives of the instantaneous squared error. These derivatives form the elements of the Hessian matrix [13], which involves the basis of a fast procedure for re-training a feed forward network following a small change in the training data. Thus, due to the several applications of the Hessian matrix, there is various approximation schemes have been used to evaluate it like the diagonal approximation [14] outer product approximation and inverse Hessian [15]. The exact evaluation of the Hessian matrix has also proposed, which is valid for a network of arbitrary feed forward topology. This method is based on an extension of the technique of the back propagation used to evaluate the first derivatives, and shares the many desirable features of it [16]. The second derivative of the error with respect to weight is obtained as conjugate descent method [17, 18]. The further improvement in conjugate descent method is also considered [19] and in the family of Quasi – Newton algorithms [20]. Further, the influence of gain was studies by few researchers [21 – 23]. The gain parameter controls the steepness of the activation function. It has been shown that a larger gain value has an equivalent effect of increasing the learning rate. Recently it has been suggested that a simple modification to the initial search direction, i.e. the gradient of error with respect to weights, can substantially improve the training efficiency. It was discovered that if the gradient based search direction is locally modified by a gain value used in the activation function of the corresponding node, significantly improvements in the convergence rates can be achieved [24].

In this present paper, we are considering a multilayer fed forward neural network with a training set of English alphabets. This neural network is trained for good generalization with generalized second derivative delta learning rule for the stochastic error. This random error is back propagated among the units of hidden layers, for the modification of the connection weights in order to minimize the error. This modification in the weights is preformed with generalized second derivative of error with respect to weights between hidden layer and output layer and also in between input and hidden layer. The neural network is trained for capturing the generalized implicit functional relationship between input and output pattern pairs. Thus, it is expected from the adaptive neural network is that it could able to recognize the individual characters from the handwritten English word of three letters. Hence, the proposed method is providing the generalized way for minimization of optimal global error which consists with instantaneous unknown minimum local errors.

2. Feed forward neural network with delta learning rule

A multi layer feed forward neural networks normally consist with the input, output and hidden layers. The processing units in the output layer and hidden layers usually contain non linear differentiable output function and the units of input layer use the linear output function. If the units in the hidden layers and in the output layer are non-linear, then the number of unknown weights or connection strengths depend on the number of units in the hidden layers, besides the number of units in the input and the output layers. Obviously, the network is suppose to use for generalization i.e. pattern mapping. Thus, the pattern mapping, problem involves determining these weights, for the given training input-output pattern pairs as shown in figure 1.

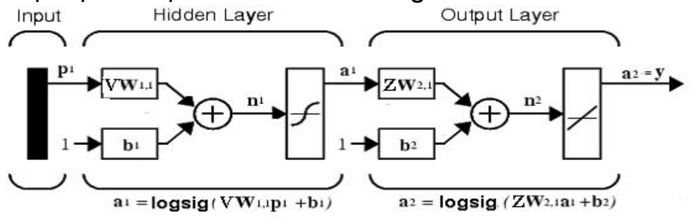

Figure 1: Architecture of the multi layer feed forward Neural Network

So far, in order to determine the weights in supervisory mode it is necessary to know the error between the derived or expected output and the actual output of the network for a given training pattern. We know the desired output only for the units in the final output layer, not for the units in the hidden layers. Thus, the same error of output layer is back propagated for the hidden layers to guide the updating or modification in the weights. Therefore, the instantaneous error can minimize by updating the weights between the input layer to hidden layer and hidden to output layer. Thus, we use the approach of gradient descent along the error surface in the weights space to adjust the weights to arrive at the optimum weight vector. The error is defined as the squared difference between the desired output and the actual output obtained at the output layer of the network due to application of an input pattern from the given input-output pattern pairs. The output has calculated using the current setting of the weights in all the layers as follows:

$$y_k = f(\sum_{i=1}^{n} Z_i W_{ik}) \qquad (2.1)$$

Where f the output function, Z_i is the output of hidden layer and W_{ik} is the weight between hidden and output layer.

And , also for hidden layer's processing unit output ;

$$Z_i = f(\sum_{j=1}^{n} V_{ij} X_j) \qquad (2.2)$$

Where X_j the output of input is layer and V_{ij} is the weight between input and hidden layer.

The instantaneous squared error for the lth pattern can represent as:

$$E^l = 0.5 \sum_k [t_k - y_k]^2 \qquad (2.3)$$

Where t_k is desired output.

Thus, for each input-output pattern pair the network has the different error. So, we have the local errors for the given input-output pattern pairs.

The weights are updated to minimize the current local error or unknown instantaneous square error for each presented input-output pattern pair. Thus, the optimum weights may be obtained if the weights are adjusted in such a way that the gradient descent is made along the total error surface. The error minimization can be shown as;

$$\partial E^l / \partial W_{ik} = [t_k - y_k] f'(y_k) z_i \qquad (2.4)$$

Weights modifications on the hidden layer can be defined as;

$$\Delta V_{ij} = \partial E^l / \partial V_{ij} = \sum_k \partial k * (\partial y_k / \partial v_{ij})[t_k - y_k] f'(y_k) z_i \quad (2.5)$$

[Let, $\partial k = [t_k - y_k]\ f'(y_k)$]

And we have

$$\Delta V_{ij} = \sum_k \partial k * W_{ik} f'(z_i) \qquad (2.6)$$

Thus the weight updates for output unit can be represented as;

$$W_{ik}(t+1) = W_{ik}(t) + \eta\ \Delta W_{ik}(t) + \alpha\ \Delta W_{ik}(t-1) \qquad (2.7)$$

Where

$W_{ik}(t)$ is the state of weight matrix at iteration t

$W_{ik}(t+1)$ is the state of weight matrix at next iteration

$W_{ik}(t-1)$ is the state of weight matrix at previous iteration.

$\Delta W_{ik}(t)$ is current change/ modification in weight matrix and α is standard momentum variable to accelerate learning process.

This variable depends on the learning rate of the network. As the network yields the set learning rate the momentum variable tends to accelerate the process.

In this process the weights are updating on each step for the instantaneous local unknown square error instead of least mean error or total error or global error for the entire training set. The determination of the total error surface cannot be known, because the set of input-output pattern pairs is large and continuous. Thus, the gradient descent on each step is obtained along the instantaneous local error surface. Hence the weights are now adjusted in a manner that the network is leading towards the minima of local error surface for the presented input-output pattern pair. Therefore to obtain the optimum weight vector for the given training set, the weights must modify in a manner that the network should leads towards the minimum of global error i.e. the expected value of the error function for all the training samples. As we can observe from the figure 1, that the output vector is of k dimension and so for we have the k-dimensional error surface. The network will lead towards the descent gradient of k- dimensional error surface. Now, another input-output pattern pair i.e. l + 1 represents and may have another error surface which is different from the first one. Therefore the weights of the network will further modified as per equation (2.5) for the error of l + 1 pattern i.e. E^{l+1} . Hence, this process will continue for every presented pattern pair and we may have the different error surfaces. Only one error surface at a one time will activate for the presented input pattern. It is easy to interpret that, every time the network tries to minimize the current local unknown error. Now, it is clear that in descent gradient learning rule weights of the network are updating only on the basis of local error rather than the global error and in order to obtain the generalized behavior the updating in the weights are required along the descent gradient of the global error or least mean square error for the entire training set.

Hence, it can realize that to obtain the good generalization for the given input output pattern pairs the weight updating should take place for the global error rather than the local error surfaces. Therefore, we can visualize a k- dimensional error surface in which we have different descent gradient corresponding to different input output pattern pairs, but only one descent gradient will activate at a one time. So, it is difficult to keep the entire local descent gradients and to search for the global one. Instead of this, we can keep the different minima points of the error corresponding to different input output pattern pairs. These minima points will distribute in the entire error surface and to trace the global minima from these local minimum points will easy and convenient. Thus, to accomplish the determination of local minimum points, we can consider the second derivative of descent gradient of the local errors.

3. Generalized Second Derivative for Feed Forward Multilayer Neural Networks:

In this section we present a generalized method for obtaining the second derivative of the descent gradient of global error with respect to weight in weight space for the good generalized behavior of the feed forward multilayer neural network for the given training set. Therefore to obtain the optimal weight vector for the feed forward neural network, the weight modification should perform for the global minimum point among the various local error minima points. Thus, the modification in the weight vector in each step is for minimizing first the local instantaneous square error and in second step is to minimize the current global or least mean square error. Now, we illustrate the generalized method for determining the second derivative of instantaneous error and further for the current global error correspond to presented input-output pattern pairs on each step. The error can obtain for the feed forward neural network as shown in fig (1) for the lth pattern from the equation (2.3) and decent gradient for the instantaneous square error as obtained from the equation (2.4). Now from these two equations we have;

$$\Delta W_{kj} = -\eta \frac{\partial E^l}{\partial W_{kj}} = -\eta \frac{\partial}{\partial W_{kj}}[\frac{1}{2}\sum_{k=1}^{K}(b_k^l - s_k^l)^2] \ldots\ldots (3.1)$$

Now, we consider the second derivative for the error as [23];

$$\frac{\partial^2 E^l}{\partial W_{kj}^2} = \frac{\partial}{\partial W_{kj}}[\frac{\partial E^l}{\partial W_{kj}}] = \frac{\partial}{\partial W_{kj}}[\frac{\partial E^l}{\partial y_k}.\frac{\partial y_k}{\partial W_{kj}}]$$

where $y_k = \sum_{j=1}^{J} W_{kj}.s_j^l$ (activation from the output layer's units)

or $$\frac{\partial^2 E^l}{\partial W_{kj}^2} = \frac{\partial}{\partial W_{kj}}[\frac{\partial E^l}{\partial y_k}.s_j^l] = \frac{\partial}{\partial y_k}[\frac{\partial E^l}{\partial W_{kj}}.s_j^l]$$

International Journal of Scientific & Engineering Research Volume 4, Issue 8, August-2013
ISSN 2229-5518

$=\frac{\partial}{\partial y_k}[\frac{\partial E^l}{\partial y_k}.(s_j^l)^2]$, since $\frac{\partial s_j^l}{\partial W_{kj}}=0$

Hence, $\frac{\partial^2 E^l}{\partial W_{kj}^2}=\frac{\partial^2 E^l}{\partial y_k^2}(s_j^l)^2$ .....(3.2)

We further extend the derivative term from the equation (3.2) as;

$$\frac{\partial^2 E^l}{\partial y_k^2}=\frac{\partial}{\partial y_k}[\frac{\partial E^l}{\partial s_k^l}.\frac{\partial s_k^l}{\partial y_k}]=\frac{\partial}{\partial y_k}[\frac{\partial E^l}{\partial s_k^l}.\dot{s}_k^l];$$

Where $\dot{s}_k^l=\frac{\partial s_k^l}{\partial y_k}$ and the non linear differentiable output signal is defined as;

$$s_k^l=f(y_k^l)=\frac{1}{1+\exp(-ky_k^l)}$$

So,we have $\frac{\partial^2 E^l}{\partial y_k^2}=\frac{\partial}{\partial y_k}[\frac{\partial E^l}{\partial s_k^l}.\dot{s}_k^l]=\frac{\partial^2 E^l}{\partial y_k \partial s_k^l}.\dot{s}_k^l+\frac{\partial E^l}{\partial s_k^l}.\frac{\partial \dot{s}_k^l}{\partial y_k}$

$$=\dot{s}_k^l[\frac{\partial}{\partial s_k^l}(\frac{\partial E^l}{\partial y_k})]+\frac{\partial E^l}{\partial s_k^l}.\partial \dot{s}_k^l$$

$$=s_k^l(1-s_k^l)[\frac{\partial}{\partial s_k^l}(\frac{\partial E^l}{\partial s_k^l}.\dot{s}_k^l)]+\frac{\partial E^l}{\partial s_k^l}.\dot{s}_k^l(1-2s_k^l)$$

$$=s_k^l(1-s_k^l)[(\frac{\partial^2 E^l}{(\partial s_k^l)^2}.\dot{s}_k^l+\frac{\partial E^l}{\partial s_k^l}.\frac{\partial \dot{s}_k^l}{\partial s_k^l})]+\frac{\partial E^l}{\partial s_k^l}\dot{s}_k^l(1-2s_k^l)$$

$$=s_k^l(1-s_k^l)[(\frac{\partial^2 E^l}{(\partial s_k^l)^2}.\dot{s}_k^l+\frac{\partial E^l}{\partial s_k^l}(1-2s_k^l)]-\sum_{k=1}^{K}(b_l^k-s_l^k)s_k^l(1-2s_k^l)$$

(For all the units of the output layer)

$$=\sum_{k=1}^{K}s_k^l(1-s_k^l)[\dot{s}_k^l-(b_l^k-s_l^k).(1-2s_k^l)]-\sum_{k=1}^{K}(b_l^k-s_l^k)s_k^l(1-s_k^l)(1-2s_k^l)]$$

$$=\sum_k s_k^l(1-s_k^l)[s_k^l(1-s_k^l)-s_l^k(1-2s_k^l)-\delta_l^k(1-2s_k^l)]$$

, where $\delta_l^k=(b_l^k-s_l^k)$

$$=\sum_k s_k^l(1-s_k^l)[s_k^l(1-s_k^l)-\delta_l^k(1-2s_k^l)-\delta_l^k(1-2s_k^l)]$$

$$=\sum_k s_k^l(1-s_k^l)[s_k^l(1-s_k^l)-2\delta_l^k(1-2s_k^l)]$$

$$\frac{\partial^2 E^l}{\partial y_k^2}=\sum_k \dot{s}_k^l[\dot{s}_k^l-2\delta_l^k(1-2s_k^l)] \quad ....(3.3)$$

So that, $\frac{\partial^2 E^l}{\partial W_{kj}^2}=\sum_k \dot{s}_k^l[\dot{s}_k^l-2\delta_l^k(1-2s_k^l)].(s_j^l)^2$ ...(3.4)

Thus, the weight adjustment can obtain corresponding to the minima point of error E^l as;

$$\Delta W_{kj}=-\eta\sum_k \dot{s}_k^l[\dot{s}_k^l-2\delta_l^k(1-2s_k^l)].(s_j^l)^2 \quad ....(3.5)$$

Correspondingly the new weight between the processing units of hidden and output layer can obtain as;

$$W_{kj}(t+1)=W_{kj}(t)-\eta\sum_k \dot{s}_k^l[\dot{s}_k^l-2\delta_l^k(1-2s_k^l)].(s_j^l)^2$$

....................(3.6)

Now, we determine the weight modification between the processing units of input layer and hidden layer of feed forward neural network as shown in fig.1, in order to minimize the same local error E^l and to obtain the local minima point of the error. Again we will consider the second derivative of the error with respect to the weight W_{ji} as;

$$\Delta W_{ji}=-\eta\frac{\partial^2 E^l}{\partial W_{ji}^2} \quad ....(3.7)$$

So, on illustration of the term $\frac{\partial^2 E^l}{\partial W_{ji}^2}$ we have;

$$\frac{\partial^2 E^l}{\partial W_{ji}^2}=\frac{\partial}{\partial W_{ji}}(\frac{\partial E^l}{\partial W_{ji}})=\frac{\partial}{\partial W_{ji}}[\frac{\partial E^l}{\partial y_j}.\frac{\partial y_j}{\partial W_{ji}}]$$

Where $y_j=\sum_{i=1}^{I}W_{ji}.a_i^l$ (activation from the hidden layer's unit)

And $a_i^l=f(a_i^l)$ is the applied input on the ith unit of input layer.

Hence, $\frac{\partial}{\partial W_{ji}}[\frac{\partial E^l}{\partial y_j}.a_i^l]=\frac{\partial}{\partial y_j}[\frac{\partial E^l}{\partial W_{ji}}.a_i^l]=\frac{\partial^2 E^l}{\partial y_j^2}.(a_i^l)^2$,

since $\frac{\partial a_i^l}{\partial W_{ji}}=0$ ....(3.8)

We further expand the derivative term from the equation (3.8) as;

$$\frac{\partial^2 E^l}{\partial y_j^2}=\frac{\partial}{\partial y_j}(\frac{\partial E^l}{\partial y_j})=\frac{\partial}{\partial y_j}[\frac{\partial E^l}{\partial s_j^l}.\frac{\partial s_j^l}{\partial y_j}]=\frac{\partial}{\partial y_j}[\frac{\partial E^l}{\partial s_j^l}.\dot{s}_j^l]$$

Where $s_j^l=f(y_j^l)=\frac{1}{1+\exp(-y_j)}$ an output signal with k=1)

So that, $\frac{\partial^2 E^l}{\partial y_j^2}=\frac{\partial}{\partial y_j}(\frac{\partial E^l}{\partial s_j^l}.\dot{s}_j^l)=\frac{\partial}{\partial y_j}[\frac{\partial E^l}{\partial y_k}.\frac{\partial y_k}{\partial s_j^l}.\dot{s}_j^l]$

$$=\frac{\partial}{\partial y_j}[\frac{\partial E^l}{\partial y_k}.W_{kj}.\dot{s}_j^l]$$

So, we have

$$\frac{\partial^2 E^l}{\partial y_j^2}=\frac{\partial}{\partial y_j}[\frac{\partial E^l}{\partial y_k}.W_{kj}.\dot{s}_j^l]$$

$$=\frac{\partial^2 E^l}{\partial y_j \partial y_k}.W_{kj}.\dot{s}_j^l+\frac{\partial E^l}{\partial y_k}\frac{\partial W_{kj}}{\partial y_j}.\dot{s}_j^l+\frac{\partial E^l}{\partial y_k}.W_{kj}.\frac{\partial \dot{s}_j^l}{\partial y_j}$$

International Journal of Scientific & Engineering Research Volume 4, Issue 8, August-2013
ISSN 2229-5518

$$=\frac{\partial}{\partial y_k}[\frac{\partial E^l}{\partial y_j}].W_{kj}.\dot{s}_j^l+\frac{\partial E^l}{\partial y_k}.W_{kj}.\frac{\partial \dot{s}_j^l}{\partial y_j}$$ Since, $\frac{\partial W_{kj}}{\partial y_j}=0$

=

$$\frac{\partial}{\partial y_k}[\frac{\partial E^l}{\partial y_j}].W_{kj}.\dot{s}_j^l-\sum_k(b_k^l-s_k^l).s_k^l(1-s_k^l).W_{kj}.s_j^l(1-s_j^l)(1-2s_j^l)$$

$$=\frac{\partial^2 E^l}{\partial y_k y_j}.W_{kj}.\dot{s}_j^l-\Delta^l \qquad \text{............(3.9)}$$

Where $\Delta^l=\sum_k(b_k^l-s_k^l).s_k^l(1-s_k^l).W_{kj}.s_j^l(1-s_j^l)(1-2s_j^l)$

Or $\frac{\partial^2 E^l}{\partial y_j^2}=\frac{\partial}{\partial y_k}[\frac{\partial E^l}{\partial s_j^l}.\frac{\partial s_j^l}{\partial y_j}].W_{kj}.\dot{s}_j^l-\Delta^l$

$$=\frac{\partial}{\partial y_k}[\frac{\partial E^l}{\partial s_j^l}.\dot{s}_j^l].W_{kj}.\dot{s}_j^l-\Delta^l$$

$$=\frac{\partial}{\partial y_k}[\frac{\partial E^l}{\partial y_k}.W_{kj}.\dot{s}_j^l].W_{kj}.\dot{s}_j^l-\Delta^l$$

$$=\frac{\partial^2 E^l}{\partial y_k^2}.W_{kj}.\dot{s}_j^l+\frac{\partial E^l}{\partial y_k}.\frac{\partial W_{kj}}{\partial y_k}.\dot{s}_j^l+\frac{\partial E^l}{\partial y_k}.W_{kj}.\frac{\partial \dot{s}_j^l}{\partial y_k}].W_{kj}.\dot{s}_j^l-\Delta^l$$

$=[\frac{\partial^2 E^l}{\partial y_k^2}.W_{kj}.\dot{s}_j^l].W_{kj}.\dot{s}_j^l-\Delta^l$, since $\frac{\partial W_{kj}}{\partial y_k}=0$ and $\frac{\partial \dot{s}_j^l}{\partial y_k}=0$

(Because the output signal of hidden layer unit is independent of the change in the activation of output layer unit)

$$=-\frac{\partial}{\partial y_k}[\sum_k(b_k^l-s_k^l).\dot{s}_k^l].W_{kj}.\dot{s}_j^l-\Delta^l$$

$$=\sum_k[(\dot{s}_k^l)^2-(b_k^l-s_k^l).\ddot{s}_k^l].W_{kj}.\dot{s}_j^l-\Delta^l$$

$$=\sum_k[(\dot{s}_k^l)^2-(b_k^l-s_k^l).\dot{s}_k^l(1-2s_k^l)].W_{kj}.\dot{s}_j^l-\Delta^l$$

$$=\sum_k\dot{s}_k^l[\dot{s}_k^l-(b_k^l-s_k^l)(1-2s_k^l)].W_{kj}.\dot{s}_j^l-\Delta^l$$

Hence, $\frac{\partial^2 E^l}{\partial y_j^2}=\sum_k\dot{s}_k^l[\dot{s}_k^l-\delta_l^k(1-2s_k^l)].W_{kj}.\dot{s}_j^l-\Delta^l$

......(3.10)

Now, from equation (3.8) we have,

$$\frac{\partial^2 E^l}{\partial W_{ji}^2}=\sum_k\dot{s}_k^l[\dot{s}_k^l-\delta_l^k(1-2s_k^l)].W_{kj}.\dot{s}_j^l.(a_i^l)^2-\Delta^l(a_i^l)^2$$

...........(3.11)

Hence, from equation (3.7) and (3.11) we have;

$$\Delta W_{ji}=-\eta[\Delta^l(a_i^l)^2-\eta\sum_k\dot{s}_k^l[\dot{s}_l^k-\delta_l^k(1-2s_k^l)].W_{kj}.\dot{s}_j^l.(a_i^l)^2$$

.....(3.12)

Thus, the new weights between the processing units of input layer and hidden layer can obtain as;

$$W_{ji}(t+1)=W_{ji}(t)-\eta\Delta^l(a_i^l)^2-\eta\sum_k\dot{s}_k^l[\dot{s}_k^l-\delta_l^k(1-2s_k^l)].W_{kj}.\dot{s}_j^l.(a_i^l)^2$$

.................(3.13)

So that, here we have obtained the weight modification for the feed forward neural network in order to minimize the local error for the presented input-output pattern pair. The second derivative of local error has been calculated separately with respect to the weights between hidden and output layers, input and hidden layer. Now, we are determining the second derivatives of the same local error with respect to the weights of input and hidden layer and hidden and output layer in combination. Thus, again we consider the local error and the gradient descent of error surface in weight space as;

$$E^l=\frac{1}{2}\sum_{k=1}^{K}[b_l^k-s_l^k]^2$$

and $$\Delta W_{h0}=-\eta\frac{\partial^2 E^l}{\partial W_{ji}\partial W_{kj}} \qquad \text{...............(3.14)}$$

Where W_{h0} represents the one weight from each hidden and output layer.

So that, $\frac{\partial^2 E^l}{\partial W_{ji}\partial W_{kj}}=\frac{\partial}{\partial W_{ji}}[\frac{\partial E^l}{\partial W_{kj}}]=\frac{\partial}{\partial W_{ji}}[\frac{\partial E^l}{\partial y_k}.\frac{\partial y_k}{\partial W_{kj}}]$

$$=\frac{\partial}{\partial W_{ji}}[\frac{\partial E^l}{\partial y_k}.s_l^j]=\frac{\partial}{\partial W_{ji}}[\frac{\partial E^l}{\partial s_k^l}.\dot{s}_k^l.s_j^l]$$

$$=\frac{\partial}{\partial s_k^l}\frac{\partial E^l}{\partial W_{ji}}.\dot{s}_k^l.s_j^l+\frac{\partial E^l}{\partial s_k^l}.\frac{\partial \dot{s}_k^l}{\partial W_{ji}}.s_j^l+\frac{\partial E^l}{\partial s_k^l}.\dot{s}_k^l.\frac{\partial s_j^l}{\partial W_{ji}}$$

$$=\frac{\partial}{\partial s_k^l}\frac{\partial E^l}{\partial W_{ji}}.\dot{s}_k^l.s_j^l+\frac{\partial E^l}{\partial s_k^l}.\frac{\partial \dot{s}_k^l}{\partial W_{ji}}.s_j^l-\sum_k(b_k^l-s_k^l).\dot{s}_k^l.\dot{s}_j^l.a_i^l$$

$$=\frac{\partial}{\partial s_k^l}\frac{\partial E^l}{\partial W_{ji}}.\dot{s}_k^l.s_j^l-\sum_k(b_k^l-s_k^l).(s_j^l)^2\dot{s}_k^l(1-2s_k^l)-\sum_k(b_k^l-s_k^l).\dot{s}_k^l.\dot{s}_j^l.a_i^l$$

$$=(\dot{s}_k^l.s_j^l)^2-\sum_k(b_k^l-s_k^l).(1-2s_k^l)(s_j^l)^2\dot{s}_k^l-\sum_k(b_k^l-s_k^l).(s_j^l)^2\dot{s}_k^l(1-2s_k^l)-\sum_k(b_k^l-s_k^l).\dot{s}_k^l.\dot{s}_j^l.a_i^l$$

$$=(\dot{s}_k^l.s_j^l)^2-\sum_k(b_k^l-s_k^l)[(1-2s_k^l)(s_j^l)^2\dot{s}_k^l+(s_j^l)^2\dot{s}_k^l(1-2s_k^l)+\dot{s}_k^l.\dot{s}_j^l.a_i^l]$$

$$=(\dot{s}_k^l.s_j^l)^2-\sum_k(b_k^l-s_k^l)[\dot{s}_k^l[2[(1-2s_k^l)(s_j^l)^2]+.\dot{s}_j^l.a_i^l]]$$

Hence, the weight change is;

$$W_{ho}=\eta\sum_k(b_k^l-s_k^l)[\dot{s}_k^l[2(1-2s_k^l)(s_j^l)^2+\dot{s}_j^l.a_i^l]-\eta(\dot{s}_k^l.s_j^l)^2$$

.............(3.15)

Hence, from the above mentioned expression we can obtain weight modification in terms of second derivative of error with respect to weights of the hidden-input layer and output-hidden layer. The weight modification has been obtained for the units of hidden layer in the terms of back propagated error and for the units of output layer in the terms of local error generated from the units of output layer. The weight modification has also been obtained for the one weight from the each hidden and output layer. Thus, for each input–output pattern pair of the training set the weight vector will incrementally update according to the learning

International Journal of Scientific & Engineering Research Volume 4, Issue 8, August-2013
ISSN 2229-5518

equation i.e.

$$W(t+1) = W(t) + \Delta W \qquad \text{………….(3.16).}$$

This process will continue for the each presented input- output pattern pairs. Here we have obtained the minimum point of each local instantaneous squared error by determining its second derivative. In this way, we have the collection of local error minimum points in k-dimensional error surface. Now we can determine the global minimum point by taking the square mean of current local error point with the current square mean of the error point i.e. the current global or total error point.

Let, initialize the global error with zero i.e. $E_g^{\min} = 0$ and determine the local instantaneous error point correspond to the presented input – output pattern pair i.e. $(a^{'l}, b^{'l}) = E^l$. The current global error point can determine as;

$$E_g^{\min} = ((E_g^{\min} + E^l)^2)/2 \qquad \text{……………………….(3.17)}$$

Now, the current weight of the network will further update as per the equation (3.5), (3.12) & (3.15) to minimize the current global error $E_g^{\min}$.

This process will continue for all the presented input-output pattern pairs of given training set and every time once the second derivative of instantaneous local error for the presented pair has obtained the $E_g^{\min}$ will modify. Thus, the minimum global error will change with every current unknown second derivative in descent direction for instantaneous local error, and the incremental weight update will preformed to minimize the current global error. Thus the process of updating of weight will accomplish two times. First time is for the second derivative of local error and furthers the second derivative of current global error. So that, in this approach the training has performed to minimize the global error. The global error has obtained in iterative dynamic fashion with the second derivative of instantaneous local errors. Hence, we have obtained the optimal weight vector for the multi layer feed forward neural network to capture generalize implicit functional relationship between input- output pattern pairs of given training set.

4 Simulation Design and Result:

In this simulation the performance of multi layer feed forward neural network trained with generalized second derivative of global error learning rule is analyzed as the good generalization for the given training set. The training set consists with English alphabets in binary form as input pattern with corresponding binary output pattern information. The generalized second derivative of instantaneous error is used to minimize the current global error which makes the network more convergent and shows the remarkable enhancement in the performance. The 1000 test sample words are presented to the vertical segmentation program which is designed in MATLAB and based on portion of average height of the words. These segmented characters are clubbed together after binarization to form training patterns for neural network. The proposed generalized second derivative delta learning rule is minimizing the current global error for each presented input output pattern pairs. The network is designed to learn its behavior by presenting each one of the 10 samples 100 times thus achieved 1000 trails. The results indicate the significant improvement in the performance of the network. To accomplish the simulation work we consider the feed forward neural network system which consists of 150x10x26 neurons in input, hidden and output layers respectively. 1000 trails have been conducted with applying different kinds of constraints for the segmentation. The constraints are based on the height of the word. The segmented characters are resized onto 15x10 binary matrixes and are exposed to 150 input neurons. The 26 output neurons correspond to 26 letters of English alphabet. The following steps have been involved for the experiments [22]:

4.1 Preprocessing: This step is considered as mandatory before segmentation and analyzing the optimal performance of the neural network for recognition. All hand written words are scanned into gray scale images. Each word is fitted into a rectangle box in order to be extracted from the document and this way they can contribute into the calculation of height and width.

4.2 The segmentation Process: The observed average height and width (H_{avg} and W_{avg}) make the basis for implementation of segmentation process. It is well observed that in cursive hand written text, the character are connected to each other to form a word at a height less than half of the H_{avg}. The following samples depict the same phenomenon:

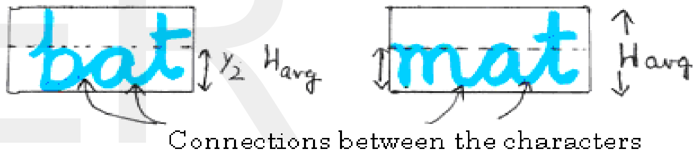

Figure 2: Connections between the characters of the cursive word.

Here, we are considering the $\frac{1}{2} * H_{avg}$ (Average of height) for deciding segmentation points. Each word is traced vertically after converting the gray scale image into binary matrix. This binarization is done using logical operation on gray intensity level as:

I = (I >= Level) Here 0<= Level <= 1 is the threshold parameter. This Level is based on the gray-scale intensity of the text in document. More intensity leads to the more threshold value. The judgment of segmentation point is based on following algorithm:

```
Algorithm: VerticalSegment (I)

1: Repeat for each column ( i ) in image matrix I starting from I (0, 0) position.
  2: Repeat for each row ( j ) element in column.
  3. Check if I (i, j ) is 0 (black pixel) and row number ( j ) > Height / 2 then
    3.1 Check if (column ( i ) < 5 ) OR ( i - last segmentation column < 5 ) then
          Process the next element.
    3.2 Else Store this segmentation point (column number i )
4. If no black pixel found in entire column then it is a segmen-
```

tation point

5. Cut the image in correspondence to segmentation points identified.

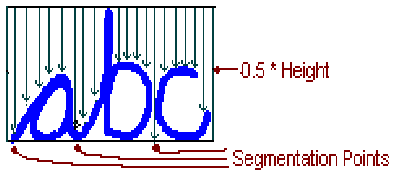

Figure 3: Vertical Segmentation Technique and Binary character

4.3 Reshape and Resizing of Characters for pattern creation: Every segmented character is first changed into a binary matrix and then resized to 15x10 matrixes using nearest neighborhood interpolation and reshaped to a 150x1 logical vector so that it can be presented to the network for learning. Such characters are clubbed together in a matrix of size (150, 26) to form the training pattern set.

4.4 Experimental Results: To analyze the performance of feed-forward neural network with conjugate descent for the pattern recognition problem the following parameters have been used in all the experiments:

Sr. No.	Parameter Name	Value
1	Learning/ Training Goal for entire network	0.0001
2	Acceptable global Error	0.0001
3	Momentum Term (α)	0.89
4	Maximum Epochs	50000
5	Initial Weights and biased term values	Randomly generated values between 0 and 1

Table 1: Parameters and their values used in all learning processes

After the segmentation technique as specified by the algorithm the following number of patterns have been obtained for the training

Segmentation Constraint	Correctly Segmented Words (Out of 1000)	Incorrect Segmented Words (Out of 1000)	Success Percentage
Height / 2	718	282	71.8 %

Table 2: Results of Vertical Segmentation Technique

Thus, out of 1000 words sample the 718 words have been correctly segmented and used as the patterns for the training of neural network. The neural network has been trained with conventional descent gradient method and from the proposed generalized second derivative of instantaneous error method with dynamic mean of the global error. The performance of the network has been analysis. The values of gradient descent and proposed generalized second derivative method are computed for each trail of learning and the mean value of all the trails of their performance has been used as the final result for the representation. The following table and graphs are exhibiting this performance analysis:

Sample	Epoch		Network Error	
	Classical Method	Second derivative Method	Classical Method	Second derivative Method
Sample1	3015	2197	0	0
Sample2	3178	2190	0	0
Sample3	3072	2252	0	0
Sample4	2852	1865	0.00343787	0
Sample5	2971	2857	0.0787574	0.000491716

Table 3: Comparison of gradient values and Error of the network

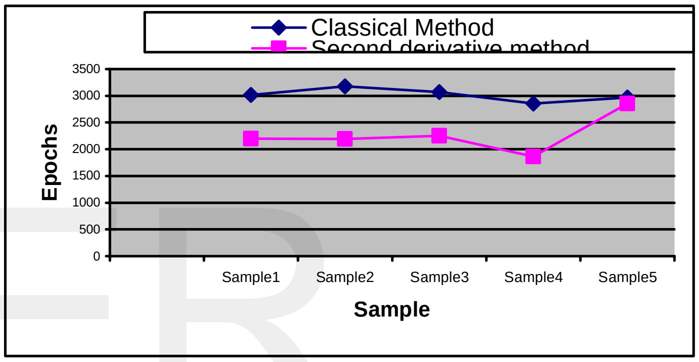

Figure 4: The Comparison chart for gradient value between Classical method and generalized second derivative methods

The results in the table 3 are representing the epochs of training and the presence of error in the network for classical gradient descent method and proposed generalized second derivative descent gradient method. The results shown here are the mean of all the trials. The proposed generalized second derivative descent gradient method for the handwritten words recognition is showing the remarkable enhancement in the performance.

5. Conclusion:

This paper is following the approach of generalized second derivative of instantaneous squared error for its minimization in the weight space corresponds to the presented input-output pattern pairs to exhibit a good generalized behavior to the fed forward neural network for the given training set. The weight modification had consider for the hidden & output layer and inputs & hidden layer, beside this the second derivative of the local errors has obtained with respect to both the weights i.e. hidden and output layer. The following observations were made for the entire discussed procedure.

1. The generalized second derivative gradient method generates the minimum of unknown instantaneous error in K-dimensional error surface. The weights have modified for each of this error. These modifications in the weights were obtained with

the generalized second derivative of this instantaneous square error. The generalized second derivative of the error has obtain with respect to the weights for output layer & hidden layer individually and also in combination.

2. The proposed method for good generalization is obtaining the point of minimum local error for every input pattern during the training in each step. Once the instantaneous local error minimum has obtained, the square mean of this local error minimum with current global error has obtained. This exhibits the current global dynamic error on each step. Further, the weights are again modified with generalized second derivative for the current global error. Thus, the network has trained for the global behavior rather than the individual local behaviors which represent the good generalized behavior of the neural network. This iterative process continues till the global error does not minimize for all the presented input –output pattern pair of training set.

3. The more experiments of complete pattern and analysis are still needed for completely verify the method. The complexity of the algorithm should also analyze and compare with the other methods. These can consider as the extended or future work.

References

[1] A. L. Blum and R. Rivest, "Training a 3- node neural network is NP-complete", Neural Networks, (5) 1 (1992) 117-128.

[2] D. R. Hush and B. G. Horne, "progress in supervised neural networks: Whats new since Lipmann", IEEE Signal Processing magazine, 10 (1993) 8-39.

[3] Williams, R. J. and Zipser, D. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. In Chauvin, Y. and Rumelhart, D. E., editors, *Back-propagation: Theory, Architectures and Applications*, chapter 13, pages 433-486. Lawrence Erlbaum Publishers, Hillsdale, N.J.

[4] Parker,D B.,(1985). Learning logic (MIT Special report TR-47). MIT Centre for Research in Computational Economics and Management Science, Massachusetts institute of technology, Cambridge, MA.

[5] FAHLMAN, S. E. 1988. An empirical study of learning speed in back-propagation networks. Tech. Rep. CMU-CS-88-162, Carnegie Mellon University, Pittsburgh, PA.

[6] Roberto Battiti, First- and second-order methods for learning: between steepest descent and Newton's method, Neural Computation, v.4 n.2, p.141-166, March 1992

[7] Jacobs, R.A. (1988) Increased rates of convergence through learning rate adaptation. *Neural Networks,* 1, 295-307.

[8] Weir M. K., "A method for self-determination of adaptive learning rates in back propagation", Neural Networks, 4 (1991), 371-379

[9] Yu X. H., Chen G. A., Cheng S. X., " Acceleration of back propagation learning using optimized learning rate momentum" Electronics Letters, 29 (14) (1993) 1288-1289.

[10] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a). *Learning internal representations by error propagation*. In D. E. Rumelhart, & J. L. McClelland (Eds.), Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1: Foundations (pp. 318–362). Cambridge, MA: MIT Press.

[11] Bishop, C. M. Neural Networks for Pattern Recognition. New York: Oxford University Press (1995).

[12] Jacobs et al.1991, Jacobs, R., Jordan, M., Nowlan, S., Hinton, G. 1991. Adaptive mixtures of local experts Neural Computation, 3, 79-87.

[13] Bishop, C. M. (1992). Exact calculation of the Hessian matrix for the multilayer perceptron. Neural Computation 4(4), 494–501.

[14] S. Backer and Y. Le Cun, "Improving the convergence of back-propagation learning with second order methods", Proceedings of the 1988 Connectionist Models Summer School, pp 2937, San Mateo, Morgan Kaufmann, 1989.

[15] Hassibi, B and stork, D.G. "second order derivatives for network pruning: optimal brain surgeon", in Advances in Neural Information processing systems, 5(164-171), 1993, San Mateo, CA: Morgan Kaufman.

[16] Buntine, W.L. and weigend A.S... "Computing second derivatives in feed-forward networks: a review". IEEE Transactions on Neural Networks, 5(3), 480-488(1993).

[17] Fletcher R. and Powell M. J. D., "A rapidly convergent descent method for nlinimization" Journal of British Computer, (1963) 163-168.

[18] Fletcher R. and Reeves R. M., "Function minimization by conjugate gradients" , Journal of Computer, 7(2) (1964) 149-160.

[19] Hestenes M. R. and Stiefel E., "Methods of conjugate gradients for solving linear systems", Journal of research NBS 49 (1952) 409-421.

[20] Huang H. Y., "A unified approach to quadratically convergent algorithms for function minimization", Journal of Optimal Theory, 5 (1970), 405-423

[21] Thimm G., Moerland F., and Fiesler E., " The interchangeability of learning rate an Gain in Back Propagation Neural Netwroks", Neural Computation 8 (2) (1996) 451-460.

[22] Dhaka V. S. and Singh M. P., "Handwritten character recognition using modified gradient descent technique of neural networks and representation of conjugate descent for training patterns", International Journal of Engineering, IJE Transaction A: Basics, 22 (2) (2009) 145-158

[23] Singh M. P., Kumar S., and Sharma N. K., "Mathematical formulation for the second derivative of backpropagation error with non-liner output function in feedforward networks", International Journal of Information and Decision Sciences, 2(4) (2010) 352-37