Volume 25 (1), pp. 69–86
http://www.orssa.org.za

ORiON
ISSN 0529-191-X
©2009

Conjugate descent formulation of backpropagation error in feedforward neural networks

Naveen Kumar Sharma[*] Sanjeev Kumar[†] Manu Pratap Singh[‡]

Received: 9 December 2008; Revised: 20 January 2009; Accepted: 18 May 2009

Abstract

The feedforward neural network architecture uses backpropagation learning to determine optimal weights between different interconnected layers. This learning procedure uses a gradient descent technique applied to a sum-of-squares error function for the given input-output pattern. It employs an iterative procedure to minimise the error function for a given set of patterns, by adjusting the weights of the network. The first derivates of the error with respect to the weights identify the local error surface in the descent direction. Hence the network exhibits a different local error surface for every different pattern presented to it, and weights are iteratively modified in order to minimise the current local error. The determination of an optimal weight vector is possible only when the total minimum error (mean of the minimum local errors) for all patterns from the training set may be minimised. In this paper, we present a general mathematical formulation for the second derivative of the error function with respect to the weights (which represents a conjugate descent) for arbitrary feedforward neural network topologies, and we use this derivative information to obtain the optimal weight vector. The local error is backpropagated among the units of hidden layers via the second order derivative of the error with respect to the weights of the hidden and output layers independently and also in combination. The new total minimum error point may be evaluated with the help of the current total minimum error and the current minimised local error. The weight modification processes is performed twice: once with respect to the present local error and once more with respect to the current total or mean error. We present some numerical evidence that our proposed method yields better network weights than those determined via a conventional gradient descent approach.

Key words: Feedforward neural networks, backpropagation learning, descent gradient, optimization.

1 Introduction

In the feedforward neural network architecture, the algorithm for modifying the weights between different interconnected layers is usually known as the backpropagation learning technique [19]. This algorithm evaluates the first derivative of the error function with

[*]CET-IILM-AHL, Knowledge park II, Greater Noida-201306, India.
[†]Department of Mathematics, Dr. B. R. Ambedkar University, Khandari, Agra, U.P., India.
[‡]Corresponding author: ICIS, Dr. B. R. Ambedkar University, Khandari, Agra, U.P., India, email: manu_p_singh@hotmail.com

respect to the weights. The error is defined as the mean square error between the desired output and the actual output of the network for any input pattern presented. The technique of backpropagation was popularized in a paper by Rumelhart, *et al.* [16]. However, a similar idea was proposed earlier by a number of researchers, including Parker [15] and Werbos [17].

Various other enhancements and modifications were also proposed for the feedforward neural network paradigm [2, 8, 12]. The term *backpropagation* is not always used consistently. For instance, the feedforward multilayer neural network architecture is sometimes also referred to as a backpropagation network. The term backpropagation is also used to describe the training of a multilayer neural network using a gradient descent approach applied to a sum-of-squares error function for the given pattern. It involves an iterative procedure of minimising this error function, by modifying the network weights during each iteration of the process. During each iteration we may distinguish between two distinct stages. In the first stage, the derivatives of the error function with respect to the weights are evaluated. In the second stage, the derivatives are then used to compute the adjustment of weights [16]. It is important to distinguish between these two stages. The first stage, namely the propagation of errors backward through the network in order to evaluate derivatives, may be applied to many other kinds of networks, not only to the multi layer perceptron. It may also be applied to error functions other than merely a simple sum-of-squares, and the evaluation of other derivatives, such as the Jacobian and Hessian matrices [3], may thus be obtained. The second stage, namely that of weight adjustment by means of these derivatives, may similarly be tackled using a verity of optimization schemes.

The derivatives of an error function with respect to weights may be obtained by propagation of errors backwards through the network, as mentioned above. That is, the weight adjustment is proportional to the negative gradient of the error with respect to the weight, where the error is the instantaneous error between the desired and actual output of the network. This instantaneous error is due to the current input pattern from the given training pattern set and becomes a sample function of a random process. Thus, the error may be assumed to be a random variable. Therefore the gradient descent method is a stochastic gradient learning method. Due to this stochastic nature, the path of the minimum along the error surface is haphazard. The error surface itself is an approximation of the true error surface determined by the entire training set of patterns. Thus, the surface may contain several local minima in addition to a global or total minimum. Consequently, the stochastic approximation of the gradient descent used during backpropagation learning need not converge. However, if the weight change is made using second derivative information of the error, then a better estimate of the optimal weight change towards the total minimum may be expected. The conjugate descent is one such method [2] where both the weight change at the previous step and the gradient at the current step are used to determine the weight change for the current step. Evaluation of the Jacobian matrix is required in this respect [4]. The elements of the Jacobian are the derivatives of the network outputs with respect to the inputs. The matrix provides a measure of the local sensitivity of outputs to changes in each of the input variables. For instance, if known errors are associated with input variables, then the Jacobian matrix allows these to be propagated through the trained network in order to estimate their contribution to the errors at the output level. In general, the network mapping represented by a trained neural network will be non-linear, and so elements of the Jacobian matrix will not be constant, but rather depend on the particular input vector used.

We may also use backpropagation to evaluate the second derivatives of the error, in order to form the Hessian matrix [3]. Several non-linear optimization algorithms used for training neural networks are, in fact, based on consideration of the Hessian matrix. The Hessian also forms the basis of a fast procedure for re-training a feedforward neural network following a small change in the training data [1]. Various approximation schemes exist for evaluating the Hessian, such as the diagonal approximation method [9], the outer product approximation method and the inverse Hessian method [6]. A method for the exact evaluation of the Hessian matrix has been also proposed, which is valid for a network of arbitrary feedforward topology. This method is based on an extension of the backpropagation technique used to evaluate first order derivatives, and shares many desirable features of it [1, 3, 13].

In the present paper, we provide a general mathematical formulation for the conjugate descent of the error function for arbitrary feedforward neural network topologies. The error is backpropagating among the units of hidden layers for modification of connection weights in order to minimise the error. The generalized delta-learning rule or backpropagation algorithm, *i.e.* the descent gradient along the unknown error surface for each of the training patterns, forms the basis for this formulation. The proposed formulation provides a general way to obtain optimal weights from the minimisation of the total or mean error which consists of combining available minimised local errors. To accomplish this we realize second order derivatives of the error and find weight modifications in order to obtain the minimum point of every unknown instantaneous local error. Hence, to obtain an optimal weight vector for a feedforward neural network, the weight modification should be performed for the total error minimum point among the various local error minima. It is necessary to modify the weights twice, *i.e.* to minimise the local instantaneous error and further to minimise the total or mean error. Generally the conjugate descent method has been observed to converge much faster than the standard backpropagation learning scheme, although there is no proof of convergence in this case due to the non-quadratic nature of the error surface [14]. We provide numerical evidence which supports this observation of superior convergence of the conjugate descent method over a standard backpropagation approach.

2 Descent gradient approach for the unknown error surface

Multilayer feedforward neural networks comprise input, output and hidden layers. Processing units in the output layer and hidden layers usually consist of nonlinear differentiable output functions, while units of the input layer typically employ linear output functions. If units in the hidden layers and in the output layer are nonlinear, then the number of unknown weights or connection strengths depend on the number of units in the hidden layers, in addition to depending on the number of units in the input and the output layers. Such networks are used for general pattern mapping, and the pattern mapping problem involves the determination of optimal connection weights for a given training set of input-output pattern pairs.

In order to determine optimal weights in a 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. However, one knows the desired output only for the units in the final output layer, not for the units in the hidden layers. Hence the combined error of output layer is backpropagated through hidden layers to guide the updating or modification of the weights. The instantaneous error may be minimised by updating weights between the input and hidden layers and between hidden and output layers. We adopt the approach of gradient descent [18] along the error surface in the weight space for adjusting the weights in order to obtain an optimal weight vector. The error is defined as sum-of-squared differences between the desired outputs and the actual outputs obtained at the output layer of the network due to application of an input pattern from a training set. The output is calculated using the current setting of weights in all layers. Thus, for each input-output pattern pair, the network produces different errors so that the local error is unknown for each given input-output pattern pair. Weights are modified in order to minimise the local error corresponding to the presented input-output pattern pair by adjusting the weights in such a way that a gradient descent is achieved along the total error surface. The total error surface is typically not known, because the set of input-output pattern pairs is usually large and presented in a continuous stream. Upon presentation of an input-output pattern pair the corresponding error may be used to calculate a descent gradient along this local error surface in order to minimise the error by adjusting weights of the network. Hence weights are adjusted in such a manner that the network approaches a minimum of the local error surface for the presented input-output pattern pairs. However, to achieve an optimal weight vector, weights should be modified in a manner that the network approaches a minimum of the total error or the mean of local minimum errors. The total error or mean error corresponds to the expected value of the error function for all training samples. Let $\{(a_\ell, b_\ell) : \ell = 1, 2, \ldots, L\}$ be the set of training pattern pairs. It is not necessary to know the entire training data set at the start, nor need the training data set be finite. The objective of the network is to determine the optimum weight update for each presentation of an input-output pattern pair.

Since the input pattern vector $[a_1^\ell, a_2^\ell, \ldots, a_I^\ell]$ is presented to the input layer and the desired output vector $[b_1^\ell, b_2^\ell, \ldots, b_K^\ell]$ for the output layer is available only at the output layer, the error between the desired output layer b_k^ℓ and the actual output vector s_k^ℓ is available only at the output layer, where $k = 1, \ldots, K$. The error for the ℓ-th pattern from each output unit is defined as

$$E^\ell = \frac{1}{2}\sum_{k=1}^{K}\left[b_k^\ell - s_k^\ell\right]^2. \tag{1}$$

The descent gradient along the error surface for the ℓ-th pattern for obtaining the increment in the weight connecting units j and k is

$$\Delta W_{jk} = -\eta\frac{\partial E^\ell}{\partial W_{jk}}, \tag{2}$$

where $\eta > 0$ is a learning rate parameter. The weight modification between the hidden and output layers and between the input and hidden layers for the ℓ-th pattern at the

$(t+1)$-th iteration are

$$W_{kj}(t+1) = W_{kj}(t) + \Delta W_{kj}(t) = W_{kj}(t) + \eta \sum_{k=1}^{K} \left[b_k^\ell - s_k^\ell \right] s_k^\ell s_j^\ell \tag{3}$$

and

$$W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji}(t) = W_{ji}(t) + \eta \sum_{k=1}^{K} \left[b_k^\ell - s_k^\ell \right] s_k^\ell W_{kj} s_j^\ell a_i^\ell, \tag{4}$$

respectively, where $\ell = 1, 2, \ldots, L$.

Thus, for the next pattern, current weights of the network are updated to accommodate the new pattern information by moving along the descent gradient of the next error surface. Hence backpropagation learning is based on the steepest descent along the error surface of the local error in the weight space. However, it is only a first order approximation of the descent as the weight change is assumed to be proportional to the negative gradient, and does not incorporate any optimization criterion. Better learning may be achieved if the supervised learning process is posed as an unconstrained optimization problem with the error function as objective function. In this case an optimal value of an increment in the weights is obtained by considering only up to second order derivatives of the error function. The resulting expression for the optimal weight change requires computation of the second order derivatives of instantaneous local errors with respect to all the weights. The additional work required to obtain these derivatives is expected to yield faster convergence than is obtained via a standard gradient descent method, but there is no guarantee that this expectation will be realised.

The incorporation of second order derivative information of the error function results in finding an optimal weight vector corresponding to different local error surfaces, all the while attempting to minimise the cumulative error corresponding to all the input-output pattern pairs. Therefore, we may visualize the process as a K-dimensional error surface in which we have different descent gradients corresponding to different input-output pattern pairs, but only one descent gradient will be active at any one time. However, it is difficult to keep track of the entire set of local descent gradients and to search for the total or mean gradient. Instead, we keep track of the different minimum points of the instantaneous local errors corresponding to different input-output pattern pairs. These minimum points will be distributed over the entire error surface, and we shall conveniently trace the mean or total errors minimum point from these local minimum points. In order to accomplish this, we require the second order derivatives of the descent gradient of the local errors.

3 Second derivative of descent gradient

The reason for requiring the second order derivatives of the descent gradient of the local errors was discussed in the previous section. In this section we determine these second order derivatives and find the weight modification in order to obtain the minimum point of every instantaneous local error. As mentioned above, the idea is that as the network trains for every presented input-output pattern pair presented, the weights are modified in order to minimise the total or mean error point. Modification of the weights therefore occurs twice: Once during minimisation of the local instantaneous error and once more during minimisation of the total or mean error.

The error produced by the feedforward neural network for the ℓ-th pattern is given in (1). The descent gradient for the error is therefore obtained from (1) and (2) as

$$\Delta W_{kj} = -\eta \frac{\partial E^\ell}{\partial W_{kj}} = -\eta \frac{\partial}{\partial W_{kj}} \left[\frac{1}{2} \sum_{k=1}^{K} (b_k^\ell - s_k^\ell)^2 \right]. \tag{5}$$

Consider the second derivative of the error,

$$\frac{\partial^2 E^\ell}{\partial W_{kj}^2} = \frac{\partial}{\partial W_{kj}} \left[\frac{\partial E^\ell}{\partial W_{kj}} \right] = \frac{\partial}{\partial W_{kj}} \left[\frac{\partial E^\ell}{\partial y_k} \frac{\partial y_k}{\partial W_{kj}} \right],$$

where $y_k = \sum_{j=1}^{J} W_{kj} s_j^\ell$ (activation from the output layer's units). Since $\frac{\partial s_j^\ell}{\partial W_{kj}} = 0$, it follows that

$$\frac{\partial^2 E^\ell}{\partial W_{kj}^2} = \frac{\partial}{\partial W_{kj}} \left[\frac{\partial E^\ell}{\partial y_k} s_j^\ell \right] = \frac{\partial}{\partial y_k} \left[\frac{\partial E^\ell}{\partial W_{kj}} s_j^\ell \right] = \frac{\partial}{\partial y_k} \left[\frac{\partial E^\ell}{\partial y_k} (s_j^\ell)^2 \right],$$

Hence,

$$\frac{\partial^2 E^\ell}{\partial W_{kj}^2} = \frac{\partial^2 E^\ell}{\partial y_k^2} (s_j^\ell)^2. \tag{6}$$

We further extend the derivative term in (6) as

$$\frac{\partial^2 E^\ell}{\partial y_k^2} = \frac{\partial}{\partial y_k} \left[\frac{\partial E^\ell}{\partial s_k^\ell} \frac{\partial s_k^\ell}{\partial y_k} \right] = \frac{\partial}{\partial y_k} \left[\frac{\partial E^\ell}{\partial s_k^\ell} \dot{s}_k^\ell \right]$$

where $\dot{s}_k^\ell = \frac{\partial s_k^\ell}{\partial y_k}$ and where the nonlinear differentiable output signal is defined as

$$s_k^\ell = f(y_k^\ell) = \frac{1}{1 + \exp(-k y_k^\ell)}.$$

We therefore have

$$\begin{aligned}
\frac{\partial^2 E^\ell}{\partial y_k^2} &= \frac{\partial}{\partial y_k} \left[\frac{\partial E^\ell}{\partial s_k^\ell} \dot{s}_k^\ell \right] \\
&= \frac{\partial^2 E^\ell}{\partial y_k \partial s_k^\ell} \dot{s}_k^\ell + \frac{\partial E^\ell}{\partial s_k^\ell} \frac{\partial \dot{s}_k^\ell}{\partial y_k} \\
&= \dot{s}_k^\ell \left[\frac{\partial}{\partial s_k^\ell} \left(\frac{\partial E^\ell}{\partial y_k} \right) \right] + \frac{\partial E^\ell}{\partial s_k^\ell} \frac{\partial \dot{s}_k^\ell}{\partial y_k}
\end{aligned}$$

and so

$$\begin{aligned}
\frac{\partial^2 E^\ell}{\partial y_k^2} &= s_k^\ell(1-s_k^\ell)\left[\frac{\partial}{\partial s_k^\ell}\left(\frac{\partial E^\ell}{\partial s_k^\ell}\dot{s}_k^\ell\right)\right] + \frac{\partial E^\ell}{\partial s_k^\ell}\dot{s}_k^\ell(1-2s_k^\ell)\\
&= s_k^\ell(1-s_k^\ell)\left[\left(\frac{\partial^2 E^\ell}{\left(\partial s_k^\ell\right)^2}\dot{s}_k^\ell + \frac{\partial E^\ell}{\partial s_k^\ell}\frac{\partial \dot{s}_k^\ell}{\partial s_k^\ell}\right)\right] + \frac{\partial E^\ell}{\partial s_k^\ell}\dot{s}_k^\ell(1-2s_k^\ell)\\
&= s_k^\ell(1-s_k^\ell)\left[\frac{\partial^2 E^\ell}{(\partial s_k^\ell)^2}\dot{s}_k^\ell + \frac{\partial E^\ell}{\partial s_k^\ell}(1-2s_k^\ell)\right] - \sum_{k=1}^{K}(b_k^\ell - s_k^\ell)s_k^\ell(1-2s_k^\ell)\\
&= \sum_{k=1}^{K} s_k^\ell(1-s_k^\ell)\left[\dot{s}_k^\ell - (b_k^\ell - s_k^\ell)(1-2s_k^\ell)\right] - \sum_{k=1}^{K}(b_k^\ell - s_k^\ell)s_k^\ell(1-s_k^\ell)(1-2s_k^\ell)\\
&= \sum_{k=1}^{K} s_k^\ell(1-s_k^\ell)\left[s_k^\ell(1-s_k^\ell) - s_k^\ell(1-2s_k^\ell) - \delta_k^\ell(1-2s_k^\ell)\right], \text{ where } \delta_k^\ell = (b_k^\ell - s_k^\ell),\\
&= \sum_{k=1}^{K} s_k^\ell(1-s_k^\ell)\left[s_k^\ell(1-s_k^\ell) - 2\delta_k^\ell(1-2s_k^\ell)\right]\\
&= \sum_{k=1}^{K} \dot{s}_k^\ell\left[\dot{s}_k^\ell - 2\delta_k^\ell(1-2s_k^\ell)\right], \qquad (7)
\end{aligned}$$

so that

$$\frac{\partial^2 E^\ell}{\partial W_{kj}^2} = \sum_{k=1}^{K}\dot{s}_k^\ell[\dot{s}_k^\ell - 2\delta_k^\ell(1-2s_k^\ell)](s_j^\ell)^2. \qquad (8)$$

Thus, the weight adjustment corresponding to the minimum point of the error E^ℓ is

$$\Delta W_{kj} = -\eta\sum_{k=1}^{K}\dot{s}_k^\ell\left[\dot{s}_k^\ell - 2\delta_k^\ell(1-2s_k^\ell)\right](s_j^\ell)^2. \qquad (9)$$

Correspondingly, the new weights between the processing units of the hidden and output layers are

$$W_{kj}(t+1) = W_{kj}(t) - \eta\sum_{k}\dot{s}_k^\ell\left[\dot{s}_k^\ell - 2\delta_k^\ell(1-2s_k^\ell)\right](s_j^\ell)^2. \qquad (10)$$

Now, we determine the weight modification between processing units of the input layer and those of the hidden layer of the feedforward neural network in order to minimise the local error E$^\ell$ and to obtain the local minimum point of the error. Again we consider the second order derivative of the error with respect to the weight W_{ji}, namely

$$\Delta W_{ji} = -\eta\frac{\partial^2 E^\ell}{\partial W_{ji}^2}. \qquad (11)$$

Illustrating for the term $\frac{\partial^2 E^\ell}{\partial W_{ji}^2}$, we have

$$\frac{\partial^2 E^\ell}{\partial W_{ji}^2} = \frac{\partial}{\partial W_{ji}}\left[\frac{\partial E^\ell}{\partial W_{ji}}\right] = \frac{\partial}{\partial W_{ji}}\left[\frac{\partial E^\ell}{\partial y_j}\frac{\partial y_j}{\partial W_{ji}}\right],$$

where $y_j = \sum_{i=1}^{I} W_{ji} a_i^\ell$ (activation from the hidden layer's unit) and where $a_i^\ell = f(a_i^\ell)$ is the input applied to the i-th unit of the input layer. Since $\frac{\partial a_i^\ell}{\partial W_{ji}} = 0$, it follows that

$$\frac{\partial}{\partial W_{ji}}\left[\frac{\partial E^\ell}{\partial y_j} a_i^\ell\right] = \frac{\partial}{\partial y_j}\left[\frac{\partial E^\ell}{\partial W_{ji}} a_i^\ell\right] = \frac{\partial^2 E^\ell}{\partial y_j^2}(a_i^\ell)^2. \tag{12}$$

We further expand the derivative term in (12) as

$$\frac{\partial^2 E^\ell}{\partial y_j^2} = \frac{\partial}{\partial y_j}\left[\frac{\partial E^\ell}{\partial y_j}\right] = \frac{\partial}{\partial y_j}\left[\frac{\partial E^\ell}{\partial s_j^\ell}\frac{\partial s_j^\ell}{\partial y_j}\right] = \frac{\partial}{\partial y_j}\left[\frac{\partial E^\ell}{\partial s_j^\ell}\dot{s}_j^\ell\right],$$

where $s_j^\ell = f(y_j^\ell) = \frac{1}{1+\exp(-y_j)}$ (an output signal with $k = 1$), so that

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

Therefore,

$$\begin{aligned}\frac{\partial^2 E^\ell}{\partial y_j^2} =& \frac{\partial}{\partial y_j}\left[\frac{\partial E^\ell}{\partial y_k} W_{kj}\dot{s}_j^\ell\right] \\ =& \frac{\partial^2 E^\ell}{\partial y_j \partial y_k} W_{kj}\dot{s}_j^\ell + \frac{\partial E^\ell}{\partial y_k}\frac{\partial W_{kj}}{\partial y_j}\dot{s}_j^\ell + \frac{\partial E^\ell}{\partial y_k} W_{kj}\frac{\partial \dot{s}_j^\ell}{\partial y_j} \\ =& \frac{\partial}{\partial y_k}\left[\frac{\partial E^\ell}{\partial y_j}\right] W_{kj}\dot{s}_j^\ell + \frac{\partial E^\ell}{\partial y_k} W_{kj}\frac{\partial \dot{s}_j^\ell}{\partial y_j},\end{aligned}$$

since

$$\begin{aligned}\frac{\partial W_{kj}}{\partial y_j} &= 0 \\ &= \frac{\partial}{\partial y_k}\left[\frac{\partial E^\ell}{\partial y_j}\right] W_{kj}\dot{s}_j^\ell - \sum_{k=1}^{K}(b_k^\ell - s_k^\ell)s_k^\ell(1-s_k^\ell)W_{kj}s_j^\ell(1-s_j^\ell)(1-2s_j^\ell) \\ &= \frac{\partial^2 E^\ell}{\partial y_k \partial y_j} W_{kj}\dot{s}_j^\ell - \Delta^\ell,\end{aligned} \tag{13}$$

where $\Delta^\ell = \sum_{k=1}^{K}(b_k^\ell - s_k^\ell)s_k^\ell(1-s_k^\ell)W_{kj}s_j^\ell(1-s_j^\ell)(1-2s_j^\ell)$. Consequently,

$$\begin{aligned}\frac{\partial^2 E^\ell}{\partial y_j^2} &= \frac{\partial}{\partial y_k}\left[\frac{\partial E^\ell}{\partial s_j^\ell}\frac{\partial s_j^\ell}{\partial y_j}\right] W_{kj}\dot{s}_j^\ell - \Delta^\ell \\ &= \frac{\partial}{\partial y_k}\left[\frac{\partial E^\ell}{\partial s_j^\ell}\dot{s}_j^\ell\right] W_{kj}\dot{s}_j^\ell - \Delta^\ell\end{aligned}$$

and so

$$\begin{aligned}\frac{\partial^2 E^\ell}{\partial y_j^2} &= \frac{\partial}{\partial y_k}\left[\frac{\partial E^\ell}{\partial y_k} W_{kj}\dot{s}_j^\ell\right] W_{kj}\dot{s}_j^\ell - \Delta^\ell \\ &= \frac{\partial^2 E^\ell}{\partial y_k^2} W_{kj}\dot{s}_j^\ell + \frac{\partial E^\ell}{\partial y_k}\frac{\partial W_{kj}}{\partial y_k}\dot{s}_j^\ell + \frac{\partial E^\ell}{\partial y_k} W_{kj}\frac{\partial \dot{s}_j^\ell}{\partial y_k} W_{kj}\dot{s}_j^l - \Delta^\ell \\ &= \left[\frac{\partial^2 E^\ell}{\partial y_k^2} W_{kj}\dot{s}_j^\ell\right] W_{kj}\dot{s}_j^l - \Delta^\ell,\end{aligned}$$

because $\frac{\partial W_{kj}}{\partial y_k} = 0$ and $\frac{\partial \dot{s}_j^\ell}{\partial y_k} = 0$ (the output signal of the hidden layer unit is independent of the change in the activation of the output layer unit). Hence

$$\begin{aligned}\frac{\partial^2 E^\ell}{\partial y_j^2} &= - \frac{\partial}{\partial y_k}\left[\sum_{k=1}^{K}(b_k^\ell - s_k^\ell)\dot{s}_k^\ell\right] W_{kj}\dot{s}_j^\ell - \Delta^\ell \\ &= \sum_{k=1}^{K}\left[(\dot{s}_k^\ell)^2 - (b_k^\ell - s_k^\ell)\ddot{s}_k^\ell\right] W_{kj}\dot{s}_j^\ell - \Delta^\ell \\ &= \sum_{k=1}^{K}\left[(\dot{s}_k^\ell)^2 - (b_k^\ell - s_k^\ell)\dot{s}_k^\ell(1-2s_k^\ell)\right] W_{kj}\dot{s}_j^\ell - \Delta^\ell \\ &= \sum_{k=1}^{K}\dot{s}_k^\ell\left[\dot{s}_k^\ell - (b_k^\ell - s_k^\ell)(1-2s_k^\ell)\right] . W_{kj}\dot{s}_j^\ell - \Delta^\ell \\ &= \sum_{k=1}^{K}\dot{s}_k^\ell\left[\dot{s}_k^\ell - \delta_k^\ell(1-2s_k^\ell)\right] W_{kj}\dot{s}_j^\ell - \Delta^\ell.\end{aligned}$$

It follows from (12) that

$$\frac{\partial^2 E^\ell}{\partial W_{ji}^2} = \sum_{k=1}^{K}\dot{s}_k^\ell\left[\dot{s}_k^\ell - \delta_k^\ell(1-2s_k^\ell)\right] W_{kj}\dot{s}_j^\ell(a_i^\ell)^2 - \Delta^\ell(a_i^\ell)^2. \tag{14}$$

Combining the expressions in (11) and (14) we therefore have

$$\Delta W_{ji} = -\eta\Delta^\ell(a_i^\ell)^2 - \eta\sum_{k=1}^{K}\dot{s}_k^\ell\left[\dot{s}_\ell^k - \delta_k^\ell(1-2s_k^\ell)\right] W_{kj}\dot{s}_j^\ell(a_i^\ell)^2. \tag{15}$$

Thus, new weights between processing units of the input layer and the hidden layer may be obtained as

$$W_{ji}(t+1) = W_{ji}(t) - \eta\Delta^\ell(a_i^\ell)^2 - \eta\sum_{k=1}^{K}\dot{s}_k^\ell\left[\dot{s}_k^\ell - \delta_k^\ell(1-2s_k^\ell)\right] W_{kj}\dot{s}_j^\ell(a_i^\ell)^2. \tag{16}$$

This relationship represents a weight modification for the feedforward neural network which minimises the local error for the presented input-output pattern pair. To prove that we indeed have a local minimal error point, we evaluate the second derivative of the error with respect to the weights. This is achieved by evaluating the second derivative separately with respect to the weights between hidden and output layers, and between input and hidden layers. Again we consider the local error in (1) and the gradient descent of the error surface in the weight space

$$\Delta W_{h_0} = -\eta \frac{\partial^2 E^\ell}{\partial W_{ji} \partial W_{kj}}, \tag{17}$$

where W_{h_0} represents the single weight from each hidden and output layer, so that

$$\begin{aligned}
\frac{\partial^2 E^\ell}{\partial W_{ji} \partial W_{kj}} &= \frac{\partial}{\partial W_{ji}} \left[\frac{\partial E^\ell}{\partial W_{kj}} \right] = \frac{\partial}{\partial W_{ji}} \left[\frac{\partial E^\ell}{\partial y_k} \frac{\partial y_k}{\partial W_{kj}} \right] = \frac{\partial}{\partial W_{ji}} \left[\frac{\partial E^\ell}{\partial y_k} s_j^\ell \right] = \frac{\partial}{\partial W_{ji}} \left[\frac{\partial E^\ell}{\partial s_k^\ell} \dot{s}_k^\ell s_j^\ell \right] \\
&= \frac{\partial}{\partial s_k^\ell} \frac{\partial E^\ell}{\partial W_{ji}} \dot{s}_k^\ell s_j^\ell + \frac{\partial E^\ell}{\partial s_k^\ell} \frac{\partial \dot{s}_k^\ell}{\partial W_{ji}} s_j^\ell + \frac{\partial E^\ell}{\partial s_k^\ell} \dot{s}_k^\ell \frac{\partial s_j^\ell}{\partial W_{ji}} \\
&= \frac{\partial}{\partial s_k^\ell} \frac{\partial E^\ell}{\partial W_{ji}} \dot{s}_k^\ell s_j^\ell + \frac{\partial E^\ell}{\partial s_k^\ell} \frac{\partial \dot{s}_k^\ell}{\partial W_{ji}} s_j^\ell - \sum_{k=1}^{K} (b_k^\ell - s_k^\ell) \dot{s}_k^\ell \dot{s}_j^\ell a_i^\ell \\
&= \frac{\partial}{\partial s_k^\ell} \frac{\partial E^\ell}{\partial W_{ji}} \dot{s}_k^\ell s_j^\ell - \sum_{k=1}^{K} (b_k^\ell - s_k^\ell)(s_j^\ell)^2 \dot{s}_k^\ell (1 - 2 s_k^\ell) - \sum_{k=1}^{K} (b_k^\ell - s_k^\ell) \dot{s}_k^\ell \dot{s}_j^\ell a_i^\ell \\
&= (\dot{s}_k^\ell s_j^\ell)^2 - \sum_{k=1}^{K} (b_k^\ell - s_k^\ell)(1 - 2 s_k^\ell)(s_j^\ell)^2 \dot{s}_k^\ell - \sum_{k=1}^{K} (b_k^\ell - s_k^\ell)(s_j^\ell)^2 \dot{s}_k^\ell (1 - 2 s_k^\ell) \\
&\quad - \sum_{k=1}^{K} (b_k^\ell - s_k^\ell) \dot{s}_k^\ell \dot{s}_j^\ell a_i^\ell \\
&= (\dot{s}_k^\ell s_j^\ell)^2 - \sum_{k=1}^{K} (b_k^\ell - s_k^\ell) \left[(1 - 2 s_k^\ell)(s_j^\ell)^2 \dot{s}_k^\ell + (s_j^\ell)^2 \dot{s}_k^\ell (1 - 2 s_k^\ell) + \dot{s}_k^\ell \dot{s}_j^\ell a_i^\ell \right] \\
&= (\dot{s}_k^\ell s_j^\ell)^2 - \sum_{k=1}^{K} (b_k^\ell - s_k^\ell) \left[\dot{s}_k^\ell \left[2 \left[(1 - 2 s_k^\ell)(s_j^\ell)^2 \right] + \dot{s}_j^\ell a_i^\ell \right] \right].
\end{aligned}$$

The weight change is therefore

$$W_{h_0} = \eta \sum_{k=1}^{K} (b_k^\ell - s_k^\ell) \left[\dot{s}_k^\ell \left[2(1 - 2 s_k^\ell)(s_j^\ell)^2 + \dot{s}_j^\ell a_i^\ell \right] \right] - \eta (\dot{s}_k^\ell s_j^\ell)^2. \tag{18}$$

This expression represents the weight modification in terms of the second derivative of the error with respect to combined weights of the hidden-input layer and output-hidden layer. The weight modification has been obtained for units of the hidden layer in the terms of the backpropagated error and for units of the output layer in terms of the local error generated from units of the output layer. The weight modification has also been obtained for the single weight from each hidden and output layer in combination. Therefore, we may represent the weight update for the entire network in terms of the updated weight vector

$$\Delta\boldsymbol{W} = \begin{bmatrix} \frac{\partial^2 E^\ell}{\partial W_{11}\partial W_{11}} & \frac{\partial^2 E^\ell}{\partial W_{11}\partial W_{12}} & \cdots & \frac{\partial^2 E^\ell}{\partial W_{11}\partial W_{1k}} \\ \frac{\partial^2 E^\ell}{\partial W_{12}\partial W_{21}} & \frac{\partial^2 E^\ell}{\partial W_{12}\partial W_{22}} & \cdots & \frac{\partial^2 E^\ell}{\partial W_{12}\partial W_{2k}} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 E^\ell}{\partial W_{21}\partial W_{11}} & \frac{\partial^2 E^\ell}{\partial W_{21}\partial W_{12}} & \cdots & \frac{\partial^2 E^\ell}{\partial W_{21}\partial W_{1k}} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 E^\ell}{\partial W_{ij}\partial W_{11}} & \frac{\partial^2 E^\ell}{\partial W_{ij}\partial W_{12}} & \cdots & \frac{\partial^2 E^\ell}{\partial W_{ij}\partial W_{1k}} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 E^\ell}{\partial W_{ij}\partial W_{j1}} & \frac{\partial^2 E^\ell}{\partial W_{ij}\partial W_{j2}} & \cdots & \frac{\partial^2 E^\ell}{\partial W_{ij}\partial W_{jk}} \end{bmatrix}_{(i+j)\times k} \tag{19}$$

in order to determine the second derivative of the local error. By presenting an input-output pattern pair for training, the weight vector will modify itself by means of the learning equation $\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \Delta\boldsymbol{W}$. This process will continue for each presented input-output pattern pair, either as a new pattern or as a repeated pattern. We are in effect finding the minimum point of each local error by determining its second derivative. In this way, we have a collection of minimum points of local errors and hence we can determine the total error or mean error minimum point by taking the mean of current local error point with respect to the current mean of error points (*i.e.* employing a dynamic mean) and it will be the current mean or total error point. We initialize the total error as zero, and determine the local error point corresponding to the presented pattern, *i.e.* $(a'^\ell, b'^\ell) = E^\ell$. The current mean error or total error point is

$$E_g^{\min} = \frac{E_g^{\min} + E^\ell}{2}, \tag{20}$$

where, initially, $E_g^{\min} = 0$.

Now, the weight vector is modified using (9), (15) and (17) in order to minimise the total error or mean error point $E_g^{\min}$.

This process is continued for all presented input-output pattern pairs, every time modifying $E_g^{\min}$ once the local error point for the presented pair has been obtained. This mean of the total error minimum point will change with every new local error minimum and the weight modification will accomplish the minimisation of the total or mean error after the minimisation of the current local error corresponding to the presented input pattern. The optimal change in the weight vector may be represented as

$$\Delta\boldsymbol{W}^{\min} = \begin{bmatrix} \frac{\partial^2 E_g^{\min}}{\partial W_{11}\partial W_{11}} & \frac{\partial^2 E_g^{\min}}{\partial W_{11}\partial W_{12}} & \cdots & \frac{\partial^2 E_g^{\min}}{\partial W_{11}\partial W_{1k}} \\ \frac{\partial^2 E_g^{\min}}{\partial W_{12}\partial W_{21}} & \frac{\partial^2 E_g^{\min}}{\partial W_{12}\partial W_{22}} & \cdots & \frac{\partial^2 E_g^{\min}}{\partial W_{12}\partial W_{2k}} \\ \vdots & \vdots & \cdots & \\ \frac{\partial^2 E_g^{\min}}{\partial W_{21}\partial W_{11}} & \frac{\partial^2 E_g^{\min}}{\partial W_{21}\partial W_{12}} & \cdots & \frac{\partial^2 E_g^{\min}}{\partial W_{21}\partial W_{1k}} \\ \vdots & \vdots & \cdots & \\ \frac{\partial^2 E_g^{\min}}{\partial W_{ij}\partial W_{j1}} & \frac{\partial^2 E_g^{\min}}{\partial W_{ij}\partial W_{j2}} & \cdots & \frac{\partial^2 E_g^{\min}}{\partial W_{ij}\partial W_{jk}} \end{bmatrix}_{(i+j)\times k}. \tag{21}$$

In this way we may determine the optimal connection strengths (weights) for the feedforward neural network to capture the relationship between all the presented input-output pattern pairs.

An algorithm for the entire method may be represented in a straight-forward manner as an extension of the backpropagation procedure for feedforward neural networks. We summarize the algorithm in the following nine steps.

1. Initialize weights and thresholds. Also initialize the total error as $E_g^{\min} = 0$ [Define $W_{ji}(t)$ and $W_{kj}(t)$, $i = 1, 2, \ldots, I$, $j = 1, 2, \ldots, J$, $k = 1, 2, \ldots, K$ to be weights from the i-th unit of the input layer to the j-th unit of the hidden layer, and weights from j-th unit of the hidden layer to the k-th unit of the output layer at time t, respectively. The parameters θ_j and θ_k are the threshold values for the j-th unit of the hidden layer and the k-th unit of the output layer, respectively. Set $\mathrm{W}_{ji}(0)$ and $\mathrm{W}_{jk}(0)$ to small random values not exceeding the threshold values].
2. Present the input $a_\ell = [a_1^\ell, a_2^\ell, \ldots, a_I^\ell]$ and the target output $b_\ell = [b_1^\ell, b_2^\ell, \ldots, b_K^\ell]$.
3. Evaluate the output of all hidden and output units, for the given input pattern, *i.e.* compute $s_j^\ell = f(\sum_{i=1}^{I} W_{ji} a_i - \theta_j)$ for the hidden layer's unit and $s_k^\ell = f(\sum_{j=1}^{J} W_{kj} s_j^\ell - \theta_k)$ for the output layer's unit. (The function $f(\cdot)$ is a sigmoidal function, which is nonlinear, differentiable and monotonically increasing).
4. Evaluate the local error
$$E^\ell = \frac{1}{2} \sum_{k=1}^{K} [l_\ell^k - s_k^\ell]^2$$
corresponding to the input pattern presented.
5. Evaluate the second derivative of the error with respect to the weights between hidden and output layers and also with respect to the weights between input and hidden layers separately as
$$\frac{\partial^2 E^\ell}{\partial W_{kj}^2} = \sum_{k=1}^{K} \dot{s}_k^\ell \left[\dot{s}_k^\ell - 2(b_k^\ell - s_k^\ell)(1 - 2 s_k^\ell) \right] (s_j^\ell)^2$$
and
$$\begin{aligned}\frac{\partial^2 E^\ell}{\partial W_{ji}^2} =& \sum_{k=1}^{K} \dot{s}_k^\ell \left[\dot{s}_k^\ell - 2(b_k^\ell - s_k^\ell)(1 - 2 s_k^\ell) \right] W_{kj} \dot{s}_j^\ell (a_i^\ell)^2 \\ & - \sum_{k=1}^{K} (b_k^\ell - s_k^\ell) s_k^\ell (1 - 2 s_k^\ell) W_{kj} s_j^\ell (1 - s_j^\ell)(1 - 2 s_j^\ell)(a_i^\ell)^2,\end{aligned}$$
respectively. Next, evaluate the second derivative of the error with respect to the weights from both hidden and output layers in combination, *i.e.*
$$\frac{\partial^2 E^\ell}{\partial W_{kj} \partial W_{ji}} = \dot{s}_k^\ell s_j^\ell - \sum_{k=1}^{K} \left[(b_k^\ell - s_k^\ell) \dot{s}_k^\ell \left[2(1 - 2 s_k^\ell)(s_j^\ell)^2 \right] + \dot{s}_j^\ell a_i^\ell \right].$$

6. Determine the weight vectors and assign these new weights to the network as the current weights. Let $W_{kj}(t+1) = W_{kj}(t)+\Delta W_{kj}(t)$ for the output layer, $W_{ji}(t+1) = W_{ji}(t)+\Delta W_{ji}(t)$ for the hidden layer, or $\boldsymbol{W}(t+1) = \boldsymbol{W}(t)+\Delta\boldsymbol{W}$, where $\Delta\boldsymbol{W}$ is given in (19). Repeat Steps 3 to 5 until the local error (E^{ℓ}) corresponding to presented input-output pattern pair (a_{ℓ}, b_{ℓ}) does not decrease.
7. Calculate the current mean of the errors as the total error point: $E_g^{\min} = (E_g^{\min} + E^{\ell})/2$, where E^{ℓ} is the local error point corresponding to the input pattern presented and $E_g^{\min}$ is the current total or mean error minimum point.
8. Modify the weights in order to minimise the total or mean error and assign new weights to the network as $\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \Delta\boldsymbol{W}$, where $\Delta\boldsymbol{W}^{\min}$ is given in (21). Calculate the output of all hidden and output units with currently assigned weights for the presented input pattern as $s_j^{\min} = f[w_j^T(t)a - \theta_j]$ (for a hidden layer unit) and $s_k^{\min} = f\left[w_k^T(t)s^{\min} - \theta_k\right]$ (for an output layer unit), respectively. Repeat Step 8 until $E_g^{\min} = \frac{1}{2}\sum_{k=1}^{K}\left[b_k^{\ell} - s_k^{\min}\right]$ does not decrease.
9. Repeat all the steps, from the Step 2, for each patterns in the training set, *i.e.* for all $\ell = 1, \ldots, L$.

4 Simulation design and result

A simulation was performed to analyze the performance of the backpropagation algorithm with changing training patterns of handwritten words consisting of three characters when using second order derivatives of the error surface to minimise the total or mean error in the feedforward neural network. It was found that use of the second order derivative of the error surface to minimise the total error yields superior network convergence. A hundred and fifty sample words were presented to the vertical segmentation program which was designed in MATLAB, based on the portion of average height of the words. These segmented characters were clubbed together after binarization to form training patterns for the neural network. The proposed conjugate gradient descent of each presented training pattern is calculated to obtain the total or mean error minimum. The network was designed to learn its behaviour from a presentation of each one of 5 samples 100 times (*i.e.* achieving 500 trials). To accomplish the simulation we considered a feedforward neural network consisting of 150, 10 and 26 neurons in the input, hidden and output layers, respectively. Five hundred experiments were conducted by applying different kinds of constraints for the segmentation of individual characters from words presented in cursive hand-written text. The constraints were based on the height of the words. The segmented characters were resized onto 15×10 binary matrices and were exposed to the 150 input neurons. The 26 output neurons correspond to letters in the English alphabet. The steps outlined in the following subsections were followed in the experiments.

4.1 Preprocessing

Preprocessing is considered mandatory before segmentation and analysis of the performance of the neural network with respect to recognition. All hand-written words were scanned into gray-scale images. Each word was fitted into a rectanglar box in order to facilitate extraction from the document and to enable calculation of height and width of characters.

4.2 The segmentation process

The observed average character height and width (H_{avg} and W_{avg}) formed the basis for implementation of the segmentation process. It has been observed that in cursive hand-written text, the characters are connected to each other to form a word at a height less than approximately half of the average maximum height. The samples in Figure 1 depict this phenomenon. We therefore considered $\frac{1}{2}H_{avg}$ (half of the average height) for deciding on segmentation points. Each word was traced vertically after converting the gray scale image into a binary matrix. This binarization was achieved using the following logical operation on the gray-scale intensity level: ($I \geq$ Level). Here $0 \leq$ Level ≤ 1 is a threshold parameter. The value of Level was based on the gray-scale intensity of the text in the document. A higher intensity leads to a larger threshold value. Determining segmentation points was carried out by means of Algorithm 1.

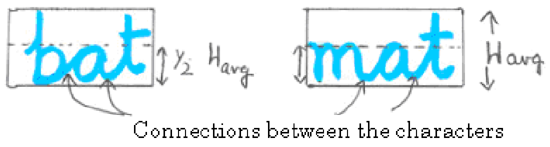

Figure 1: *Connections between the characters of a word in cursive hand-written text.*

Algorithm 1 `VerticalSegment` $\boldsymbol{I}$

```
1:  for all columns $i$ in image matrix $\boldsymbol{I}$ starting from position $\boldsymbol{I}(0,0)$ do
2:    for all rows $j$ do
3:      if $\boldsymbol{I}(i,j) = 0$ (black pixel) and row number $j >$ Height/2 then
4:        if (column $i < 5$) or ($i-$last segmentation column $< 5$) then
5:          Process the next element
6:        else
7:          Store this segmentation point (column number $i$)
8:        end if
9:      else if no black pixel found in entire column then
10:       $i$ is a segmentation point
11:       Cut the image according to the segmentation points identified
12:     end if
13:   end for
14: end for
```

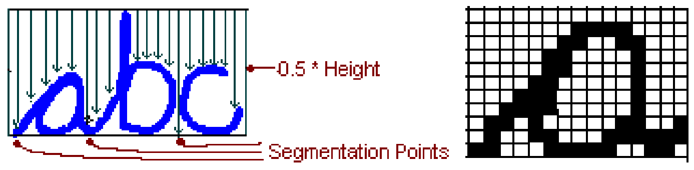

Figure 2: *Vertical segmentation technique and binary representation of a character.*

4.3 Reshaping and resizing of characters for pattern creation

As mentioned, every segmented character was first converted into a binary matrix and then resized to a 15×10 matrix using nearest neighborhood interpolation [20]. This matrix was then reshaped to a 150×1 logical vector so that it could be presented to the network for learning. These characters were clubbed together in a matrix to form the training pattern set.

4.4 Experimental results

In order to analyze the performance of the feedforward neural network with conjugate descent for the character recognition problem, the parameter values in Table 1 were used in all the experiments.

Sr. No.	Parameter Name	Value
1	Learning/ Training Goal for entire network	0.01
2	Acceptable Error	0.001
3	Classical Momentum Term (α)	0.90
4	Modified Momentum Term(β)	0.05
5	Maximum Epochs	50 000
6	Initial Weights and biased term values	Randomly generated values between 0 and 1

Table 1: *Parameters and their values used in the learning processes.*

After applying the segmentation technique, as specified in Algorithm 1, the results in Table 2 were obtained.

Segmentation Constraint	Correctly Segmented Words (Out of 600)	Incorrectly Segmented Words (Out of 600)	Success Percentage
$\frac{1}{2}H_{avg}$	427	173	71.16 %

Table 2: *Results of vertical segmentation technique.*

Thus, out of 600 words, a total of 427 words were segmented correctly and used as the patterns for the training of the neural network. The network was trained using both the conventional descent gradient method and the proposed conjugate descent method employing a dynamic mean of the total error. The values of the gradient descent and the proposed conjugate descent were computed for each learning trial and the mean values of the resulting performance is presented in Table 3 and in Figures 3–4.

It may be concluded from the table that the proposed method of conjugate descent for hand-written word recognition shows a remarkable enhancement in performance over the conventional method.

Sample	Gradient		Network Error	
	Classical Method	Proposed Method	Classical Method	Proposed Method
Sample 1	1 981 400	2 029 280	0	0
Sample 2	5 792 000	2 021 500	0	0
Sample 3	7 018 400	1 723 310	0	0
Sample 4	1 173 900	843.094	0.003 437 87	0
Sample 5	62 263.9	21 897.8	0.007 875 74	0.004 917 16

Table 3: *Comparison of gradient values and errors in the network for five samples.*

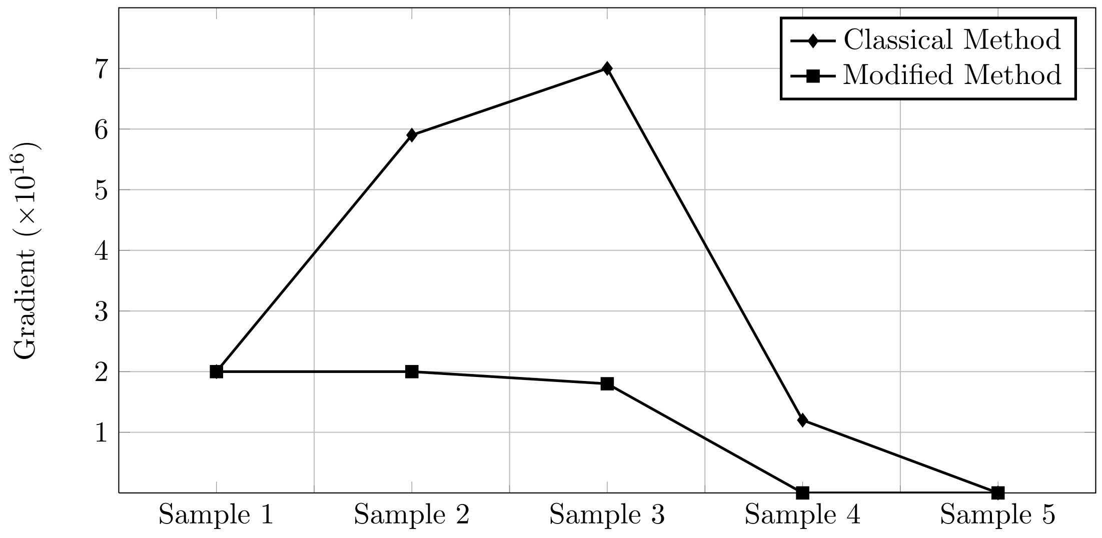

Figure 3: *Comparison of descent gradient and proposed conjugate descent methods.*

5 Conclusion

The method discussed in this paper uses second order derivative information to minimise the local error in the weight space, based on input-output pattern pairs presented to the feedforward network. Weights in the network are repeatedly modified in order to minimise the local error for each such pair. There are separate local errors for each different input-output pattern pair presented. The descent gradient of the local error in the weight space gives only the surface of error minimisation. The second derivative of the error generates the minimum point in the error surface. Hence, weights are modified to account for each of the local errors. This modification is achieved via the second derivative of the local error which is obtained with respect to weights for the output and hidden layers individually and also in combination. In this way a difference matrix is generated for the weight modification. The process of weight modification is iterative, and continues until the minimum of the local error is obtained.

At each iteration we attempt to find optimal weights for the network. These weights are responsible for producing the behaviour of the network for the test patterns. However, there is no guarantee of optimal behaviour for all patterns, because weights are modified

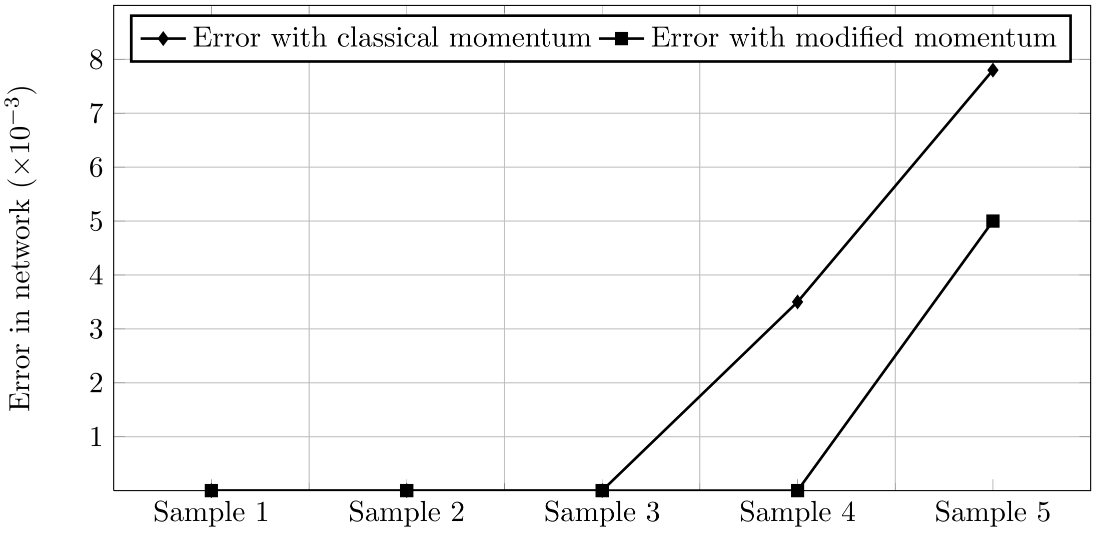

Figure 4: *Comparison of the minimised error for the descent gradient and proposed conjugate descent methods.*

for an unknown local error surface. Once the local error minimum point has been obtained, the mean of this local error minimum point provides a measure of the current total error or mean error point among the existing local error minima. Thereafter, the weights are again modified for the current total or mean error. Thus, the network is trained for the global or mean behaviour rather than for the individual local behaviours. This iterative process continues until the total or mean error does not decrease for all the input pairs presented in the training set.

The proposed conjugate descent method accelerates the process of convergence and yields better network performance. Despite the significant reduction of network error shown in Figure 5 when using the conjugate descent approach over the traditional approach for our five sample recognition experiments, we still cannot claim any guarantee or general statement with respect to the convergence of this method. Although we have the different local error minimum points corresponding to the pattern pairs presented as input, these local error minima cannot be visualized simultaneously. If the local error corresponding to a pattern has been minimised, there is a possibility that another previously obtained error minimum may change and hence that these errors are no longer minimised.

Further experimentation is required before a general conclusion may be reached with respect to the convergence rate. The complexity of the algorithm should also be analysed and compared with those of other methods. These avenues of investigation may be pursued in future work.

References

[1] Backer S & Cun YL, 1989, *Improving the convergence of backpropagation learning with second order methods*, Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufman, San Mateo (CA).

[2] Battiti R, 1992, *First- and second-order methods for learning: Between steepest descent and Newton's method*, Neural Computation, **4(2)**, pp. 141–166.

[3] Bishop CM, 1991, *A fast procedure for retraining the multilayer perceptron*, International Journal of Neural Systems, **2(3)**, pp. 229–236.

[4] Bishop CM, 1992, *Exact calculation of the Hessian matrix for the multilayer perceptron*, Neural Computation, **4(4)**, pp. 494–501.

[5] Bishop, CM 1995, *Neural networks for pattern recognition*, Oxford University Press, New York (NY).

[6] Buntine WL & Weigend AS, 1993, *Computing second derivatives in feedforward networks: A review*, IEEE Transactions on Neural Networks, **5(3)**, pp. 480–488.

[7] Casey RG & Lecolient E, 1996, *A survey of methods and strategies in characters segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, **18**, pp. 690–706.

[8] Fahlman SE, 1988, *An empirical study of learning speed in backpropagation networks*, Technical report CMU-CS-88-162, Carnegie Mellon University, Pittsburgh (PA).

[9] Hassibi B & Stork DG, 1993, *Second order derivatives for network pruning: Optimal brain surgeon*, Advances in Neural Information Processing Systems, **5**, pp. 164–171.

[10] Haykin S, 1994, *Neural networks: A comprehensive foundation*, Macmillan College Publishing Company Inc., New York (NY).

[11] Jacobs RA, 1988, *Increased rates of convergence through learning rate adaptation*, Neural Networks, **1**, pp. 295–307.

[12] Jacobs R, Jordan M, Nowlan S & Hinton G, 1991, *Adaptive mixtures of local experts*, Neural Computation, **3**, pp. 79–87.

[13] Kramer, AH & Sangiovanni-Vincentelli A, 1989, *Efficient parallel learning algorithms for neural networks*, Advances in Neural Information Processing Systems, **1**, pp. 40–48.

[14] Mandic DP, Hanna AI & Razaz M, 2001, *A normalized gradient descent algorithm for nonlinear adaptive filters using a gradient adaptive step size*, IEEE Signal Processing Letters, **11**, pp. 1–3.

[15] Parker DB, 1985, *Learning logic*, MIT Special Report TR-47, MIT Center for Research in Computational Economics and Management Science, Massachusetts Institute of Technology, Cambridge (MA).

[16] Rumelhart DE, Hinton GE & Williams RJ, 1986, *Learning internal representations by error propagation*, pp. 318–362 in Rumelhart DE & McClelland JL (Eds), *Parallel distributed processing: Explorations in the microstructure of cognition*, MIT Press, Cambridge (MA).

[17] Werbos PJ, 1974, *Beyond regression: New tools for prediction and analysis in the behavioral sciences*, PhD dissertation, Harvard University, Cambridge (MA).

[18] Widrow B & Michael AL, 1990, *30 years of adaptive neural networks: Perceptron, madaline and backpropagation*, Proceedings of the IEEE, **78**, pp. 1415–1441.

[19] Williams RJ & Zipser D, 1995, *Gradient-based learning algorithms for recurrent networks and their computational complexity*, pp. 433–486, in Chauvin Y & Rumelhart, DE (Eds), *Backpropagation: Theory, architectures and applications*, Lawrence Erlbaum Publishers, Hillsdale (NY).