MICROCOMPUTER APPLICATIONS

VOLUME 14, No. 2, 1995 *An Official Journal of the International Society for Mini and Microcomputers*

ISSN 0820-0750 *Published by* ISMM ANAHEIM • CALGARY • ZURICH

Microcomputer Applications, Vol. 14, No. 2, 1995

LEARNING ALGORITHMS FOR FEEDFORWARD NEURAL NETWORKS BASED ON CLASSICAL AND INITIAL-SCALING QUASI-NEWTON METHODS

Homayoon S.M. Beigi* and C. James Li**

Abstract

This paper describes a set of feedforward neural network learning algorithms based on classical quasi-Newton optimization techniques which are demonstrated to be up to two orders of magnitude faster than backward-propagation. Then, through initial scaling of the inverse Hessian approximate, which makes the quasi-Newton algorithms invariant to scaling of the objective function, the learning performance is further improved. Simulations show that initial scaling improves the rate of learning of quasi-Newton-based algorithms by up to 50%. Overall, more than two to three orders of magnitude improvement is achieved compared to backward-propagation. Finally, the best of these learning methods is used in developing a small writer-dependent online handwriting recognizer for digits (0 through 9). The recognizer labels the training data correctly with an accuracy of 96.66%.

Key Words

Learning, neural networks, quasi-Newton methods, unconstrained optimization, handwriting, recognition

List of Acronyms

BFGS	Broyden-Fletcher-Goldfarb-Shanno
BR.	Broyden
DFP	Davidon-Fletcher-Powell
Ex/LS	Exact Line Search
FLOP	Floating Point Operation
GRI	Greenstadt I
GRII	Greenstadt II
Inex/LS	Inexact Line Search
NN	Neural Network
PNR	Projected-Newton-Raphson
PRII	Pearson II
PRIII	Pearson III
SD	Steepest Descent
XOR	Exclusive-OR

* IBM, T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598, email: beigi@watson.ibm.com

** Mechanical Engineering Department, Rensselaer Polytechnic Institute, Troy, New York 12180, email: lic3@rpi.edu

(paper no. 307-589)

1. Introduction to Neural Network Learning

Neural net (NN) models have been studied for many years with the hope that the superior learning and recognition capability of the human brain could be emulated by man-made machines. Similar massive networks, in the human brain, make possible the complex pattern and speech recognition capabilities of humans. In contrast to the Van Neumann computers which compute sequentially, neural nets employ huge parallel networks of many densely interconnected computational elements called neurons. Neural networks have been used in many different applications such as adaptive and learning control, pattern recognition, image processing, signature recognition, signal processing and speech recognition, financial problems, etc.

A neuron is the basic computational unit in a neural network which sums a number of weighted inputs and passes the sum through a non-linear activation function. Multi-layer neural networks (Figure 1) consist of a large number of neurons. Before a neural network can be used for any purpose, the weights connecting inputs to neurons and the parameters of the activation fucntions of neurons should be adjusted so that outputs of the network will match desired values for specific sets of inputs. The methods used for adjusting these weights and parameters are usually referred to as learning algorithms.

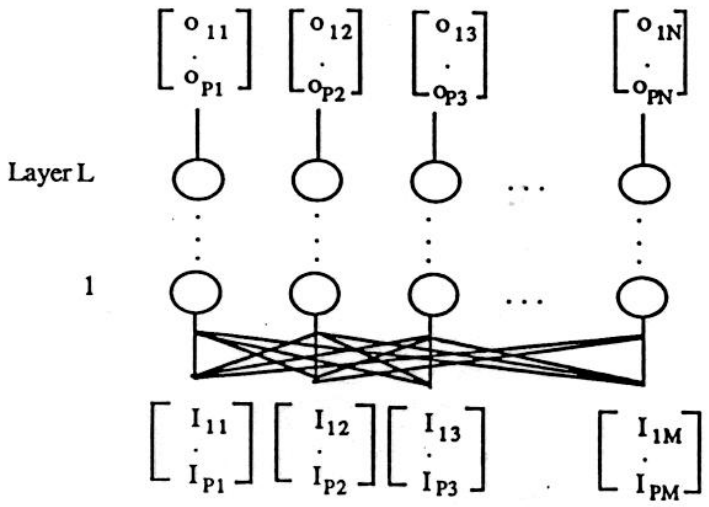

Figure 1. General multi-layer feed-forward neural network.

Rumelhart et al. [1] (also see [2, 3]) reintroduced the back-propagation learning algorithm to the community in 1986. In this algorithm, a so-called generalized delta rule is used to calculate an approximate gradient vector to facilitate a steepest descent search which minimizes the sum of squared error between the NN output and the desired output. However, as demonstrated by simulations performed by Rumehlart et al., low rates of convergence were seen in practically every problem. Lippman [4] states, "One difficulty noted by the Back-Propagation algorithm is that in many cases the number of presentations of training data required for convergence has been large (more than 100 passes through all the training data)."

In the literature, several methods [4–5] have been proposed to improve the rate of convergence of learning by making very limiting assumptions such as linearity for multi-layer networks. In addition, other more practical methods have been proposed for speeding the convergence of the back-propagation technique [6–9]. Some gradient-free techniques have also been proposed in the literature which address some practical issues arising in specific problems. These are problems that require small networks when no knowledge is available about the activation functions and connectivity of some parts of the networks [10].

The back-propagation technique, as described by Rumelhart et al., is an approximation to the steepest descent technique. In general, steepest descent techniques perform well while away from local minima and require many itertions to converge when close to the minima. On the other hand, Newton's method usually converges fast in the vicinity of minima. In addition, Newton's minimization technique elegantly handles functions with ill-conditioned Hessian matrices [11]. It would be desirable to take advantage of the properties of steepest descent when the state is far from a minimum and then to use Newton's method in the vicinity of the minimum.

To use Newton's method, the gradient and the matrix of second partial derivatives (Hessian) matrix must be evaluated. In general, two difficulties have prohibited the use of Newton's method for neural network learning: (1) the complexity of the evaluation of the Hessian, and (2) the inversion of the Hessian. One way to alleviate these difficulties is to use a momentum method which would approximate the diagonal elements of the Hessian matrix and would remain ignorant of the off-diagonal elements [12].

On the other hand, the class of quasi-Newton methods provides a better solution to the problem by providing an iterative estimate for the inverse of the Hessian matrix. If one selects the initial estimate to be an identity matrix, the method, at beginning, acts like the steepest descent and gradually changes into Newton's method as the estimate approaches the inverse of the true Hessian. This paper will first review classical quasi-Newton methods and will develop neural network learning methods based on these minimization techniques to increase the speed and accuracy of learning. Then, these newly developed learning methods will be evaluated through simulations.

Upon providing an initial estimate for the inverse of the Hessian matrix, quasi-Newton methods provide a direction for the minimization of the objective function and update the inverse Hessian based on the function and gradient values at two consecutive points. Specifically, the family of quasi-Newton methods explored in this study is the Broyden family of quasi-Newton methods.

In addition to the Broyden family of quasi-Newton methods, this paper also investigates an approach that scales the initial estimate of the inverse Hessian matrix to make the methods invariant under objective function scaling. This method was proposed by Shanno and Phua [13] and improves the speed and accuracy of the members of the Broyden family.

In Section 2, we will formulate the neural network learning problem as a minimization problem. Section 3 applies quasi-Newton methods to develop fast, new learning algorithms for neural networks. In Section 4 learning algorithms based on self-scaling are developed to speed up learning even further. Sections 5 and 6 discuss and conclude the simulation results conducted for these new learning algorithms.

In Section 5, the best of these learning methods is used in developing a small writer-dependent online handwriting recognizer for digits (0 through 9). The accuracy of classifying the training data is presented.

2. Objective Function and Gradient Vector Formulation for Learning in Feed-Forward Neural Networks

The objective of learning for neural networks is to minimize the output error of the output layer (layer L in Figure 1) of a neural network over a set of P input-output patterns.

Define,

$l \in [1, L]$ (Layer number in the network)
$n_l \in [1, N_l]$ (Neuron number in layer l)
$p \in [1, P]$ (Pattern index)
ω^l_{nm} (Weight of the mth input to neuron n in layer l)
o^l_{pn} (Output of neuron n in layer l for input pattern p)
t_{pn} (Desired output of neuron n in layer L)
i_{pm} (Input m of pattern p to the network)

Then, the objective of learning becomes minimization of the following objective function:

$$\text{Minimize } E = \sum_{p=1}^{P} E_p \tag{2.1}$$

where

$$E_p = \sum_{n_L=1}^{N_L} (o^L_{pn_L} - t_{pn_L})^2 \tag{2.2}$$

Let us define a few variables and eventually a state vector which would include all the variables to be adjusted to minimize E,

Activation function parameter vector for level l:

$$\phi^{lT} = [\phi^l_1, \phi^l_2, \ldots, \phi^l_{N_l}]$$

Vector of intercellular weights to neuron n_l at layer l:

$$\omega_{n_l}^{lT} = [\omega_{n_l 1}^{l}, \omega_{n_l 2}^{l}, \ldots, \omega_{n_l N_{l-1}}^{l}]$$

Supervector of intercellular weights of level l:

$$\omega^{lT} = [\omega_1^{lT}, \omega_2^{lT}, \ldots, \omega_{N_l}^{lT}]$$

State vector for level l:

$$x^{lT} = [\phi^{lT}, \omega^{lT}]$$

Super state vector:

$$x^{T} = [x^{1T}, x^{2T}, \ldots, x^{LT}]$$

The gradient vector, g, and the matrix of second partial derivatives (Hessian matrix), G, for the objective function are defined as follows:

$$g = \nabla_x E = \sum_{p=1}^{P} \nabla_x E_p \tag{2.4}$$

and

$$G = \nabla_x^2 E = \sum_{p=1}^{P} \nabla_x^2 E_p \tag{2.5}$$

For detailed formulation of the gradient and the Hessian, refer to [14].

3. Gradient-Based Minimization Algorithms

This section describes how minimization of an objective function can be achieved by a set of different algorithms.

3.1 Newton's Minimization Technique

Write the Taylor series expansion of E for the minimum x^* about a point x_k, and let $\Delta x_k = x^* - x_k$.

$$E(x^*) = E(x_k) + \nabla_x^T E|_{x_k} \Delta x_k + \frac{1}{2} \Delta x_k^T \nabla_x^2 E|_{x_k} \Delta x_k + O(\Delta x_k^3) \tag{3.1}$$

By using the previously defined g and G, and a subscript k to denote the point where they are evaluated, (3.1) becomes

$$E(x^*) = E(x_k) = g_k^T \Delta x_k + \frac{1}{2} \Delta x_k^T G_k \Delta x_k + O(\Delta x_k^3) \tag{3.2}$$

If the x^* and x_k are close to each other, Δx_k is small, and, therefore, one may disregard the higher than second-order terms and thus approximate E with a quadratic function:

$$E(x^*) \approx E(x_k) + g_k^T \Delta x_k + \frac{1}{2} \Delta x_k^T G_k \Delta x_k \tag{3.3}$$

Note that at a minimum of $E(x^*)$, $\nabla_x E|_{x^*}$ has to be zero. Keeping the current state x_k constant and then taking the gradients of both sides of (3.3), one has

$$\nabla_x E|_{x^*} \approx g_k + G \Delta x_k \tag{3.4}$$

Setting the gradient of E at x^* equal to zero gives

$$g_k + G \Delta x_k \approx 0 \tag{3.5}$$

or

$$\Delta x_k \approx -G_k^{-1} g_k \tag{3.6}$$

The above equation means that the minimum of a quadratic function can be found in one step. However, since E is generally not a quadratic function, a line search is usually performed along the direction suggested by (3.6) to determine the minimum along this direction. This results in

$$x_{k+1} = x_k + \lambda_k^* s_k \tag{3.7}$$

where λ_k^* is the optimum step size in the direction s_k found by a line search method, and s_k is

$$s_k = -\frac{G_k^{-1} g_k}{\|G_k^{-1} g_k\|} \tag{3.8}$$

Newton's method provides quadratic convergence and is very efficient in the vicinity of the minima. In order for E to always descend in value, the matrix G^{-1} should be positive definite at all times, which is not the case for general objective functions except the quadratic ones. There are various methods to keep the approximation of the inverse Hessian matrix (G^{-1}) positive definite. Examples include Greenstadt's method [15], Marquardt [16], Levenberg [17], and Goldfeld, Quandt and Tratter's alternative [18].

There are a couple of reasons why Newton's method cannot be easily used for neural network learning. First, for networks with a small number of neurons, it might be feasible to find the analytical expression for their Hessian matrices. However, for larger networks this will be very difficult, if not impossible. Even if the Hessian matrix is available, it would be prohibitively expensive to compute the inverse of it for networks other than very small ones. These limitations are reasons for looking at the following alternatives which can alleviate the aforementioned probems and still achieve a super-linear rate of convergence.

3.2 Quasi-Newton Methods

From equation (3.7), we can write the following generalized recursive algorithm to update the state vector such that a minimum E will be approached:

$$x_{k+1} = x_k - \lambda_k^* H_k \nabla_x E(k) \tag{3.9}$$

where λ^* is a scalar step size, and H_k is a square symmetric matrix. Depending on the choice of H_k, different optimization algorithms will result. If H_k is the identity (I) matrix, the method is the steepest descent technique which provides linear convergence. If H_k is the inverse of

the Hessian matrix (G^{-1}), the method is the Newton minimization technique which provides quadratic convergence.

Instead of using the real inverse-Hessian, quasi-Newton methods use an approximation to the inverse-Hessian provided by an iterative updating scheme. Quasi-Newton methods usually start with an initial guess of the inverse-Hessian matrix such as the identity matrix. Different updates for H_k are then used, leading to different types of quasi-Newton methods. Updates are done recursively in different directions of the inverse-Hessian space, based on the information obtained from the function and its gradient. Depending on whether these updates are done in one or two directions at a time, rank one or rank two methods are generated. Those quasi-Newton methods which retain a positive definite H_k are called variable metric methods. Not all quasi-Newton methods are variable metric updates.

It can be easily shown that the following equation is true for a quadratic function if H is the exact inverse Hessian:

$$H\Delta g_k = \Delta x_k \tag{3.10}$$

Normally, quasi-Newton methods try to update their approximation to H to satisfy the following equation:

$$H_{k+1}\Delta g_k = \Delta x_k \tag{3.11}$$

This relationship is referred to as the quasi-Newton condition.

In 1959, Davidon [19] introduced the idea of quasi-Newton methods. In 1965, Barnes [20] and Broyden [21] independently introduced a method for solving a set of simultaneous linear equations of the same form as equation (3.10). Barnes' equation, (3.12), is a more general one and Broyden's method is a special case of it.

$$\Delta H_k = \frac{(\Delta x_k - H_k\Delta g_k)z_k^T}{z_k^T\Delta g_k} \tag{3.12}$$

where z_k is a direction in which the update to H_k is done

3.2.1 Rank-One Updates to Inverse Hessian

A rank-one update to the inverse Hessian matrix H_k would mean the following:

$$H_{k+1} = H_k + \alpha uu^T$$

where u is a direction of update. Setting z_k in equation (3.12) equal to the error of equation (3.11) will produce Broyden's rank-one update:

$$\Delta H_k = \frac{(\Delta x_k - H_k\Delta g_k)(\Delta x_k - H_k\Delta g_k)^T}{(\Delta x_k - H_k\Delta g_k)^T\Delta g_k} \tag{3.13}$$

This update was introduced by Broyden [22], Davidon [23], and others [24, 25] independently. Let N denote the dimension of the state vector x. This update has the property that if $\Delta x_1, \Delta x_2, \ldots, \Delta x_N$ are linearly independent, then at $k = N+1$, $H_k = G^{-1}$ for quadratic functions.

Another important feature of Broyden's update is that λ_k^* of equation (3.13) does not necessarily have to minimize E in the s_k direction. As long as λ_k^* is such that H_{k+1} will not become singular and the denominator of (3.13) is not zero, any λ_k^* could be used in conjunction with the Broyden update. However, if the objective function is non-quadratic, as in the case of neural networks, the following undesirable aspects of the Broyden update emerge:

1. H_k may not retain its positive definiteness, in which case it is necessary to use one of the methods in section 3.2, such as Greenstadt's method, to force this matrix to be positive definite.

2. The correction ΔH_k of equation (3.13) may sometimes become unbounded (sometimes even for quadratic functions, due to round-off errors).

3. If Δx_k given by equation (3.13) is by chance in the same direction as Δx_{k-1}, then H_{k+1} becomes singular or undetermined. Therefore, if

$$H_k\Delta g_k = \Delta x_k \tag{3.14}$$

or

$$(H_k\Delta g_k - \Delta x_k)^T\Delta g_k = 0 \tag{3.15}$$

then H_{k+1} should be made equal to H_k.

3.2.2 Pearson's Updates

Pearson [26] introduced other directions of update for the projection of the error of equation (3.11). In his No. 2 method, he proposed $z_k = \Delta x_k$ in equation (3.12). Namely

$$\Delta H_k = \frac{(\Delta x_k - H_k\Delta g_k)\Delta x_k^T}{\Delta x_k^T\Delta g_k} \tag{3.16}$$

His No. 3 method used the other possibility, that is $z_k = H_k\Delta g_k$.

$$\Delta H_k = \frac{(\Delta x_k - H_k\Delta g_k)(H_k\Delta g_k)^T}{(H_k\Delta g_k)^T\Delta g_k} \tag{3.17}$$

Pearson's methods do not guarantee positive definiteness of the inverse Hessian Approximate and usually lead to ill-conditioned matrices. Therefore, it is a good idea to reset the inverse Hessian approximation to Identity, every N iterations (i.e., $H_{(Nt)} = I$ for $t = 0, 1, 2, \ldots 0$.

3.2.3 Rank-Two Updates

The rank-one updates do not leave much freedom for choosing the directions of update. This motivated the formulation of rank-two updates such that the objective is still to satisfy relation (3.11) at every step k. The general rank two updates are of the following form:

$$H_{k+1} = H_k + \alpha uu^T + \beta vv^T \tag{3.18}$$

where the directions u and v and scaling factors α and β are different for various methods. It would be a good idea to update these directions relative to Δx_k and $H_k \Delta g_k$. This would give the following general update:

$$\Delta H_k = \alpha \frac{\Delta x_k y^T}{y^T \Delta g_k} + \beta \frac{H_k \Delta g_k z^T}{z^T \Delta g_k} \tag{3.19}$$

A natural choice to retain symmetry is to pick $\alpha = 1$, $\beta = -1$, $y = \Delta x_k$, and $z = H_k \Delta g_k$. This will generate the Davidon-Fletcher-Powell (DFP) [27] update

$$\Delta H_k = \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T \Delta g_k} - \frac{H_k \Delta g_k \Delta g_k^T H_k^T}{\Delta g_k^T H_k^T \Delta g_k} \tag{3.20}$$

This algorithm works well, in general, if g_k is calculated with minimal error and H_k does not become ill-conditioned. Define

$$A_i = \frac{\Delta x_i \Delta x_i^T}{\Delta x_i^T \Delta g_i} \tag{3.21}$$

and

$$B_i = \frac{H_i \Delta g_i \Delta g_i^T H_i^T}{\Delta g_i^T H_i^T \Delta g_i} \tag{3.22}$$

Then it can be proved that [27]

$$\sum_{i=0}^{k-1} A_i \rightarrow H \text{ as } k \rightarrow N \tag{3.23}$$

and

$$\sum_{i=0}^{k-1} B_i \rightarrow H_0 \text{ as } k \rightarrow N \tag{3.24}$$

and therefore

$$H_k \rightarrow H \text{ as } k \rightarrow N \tag{3.25}$$

for a quadratic function.

An important property of this update is that if $\Delta x_k^T \Delta g_k > 0$ for all k, then the approximate inverse Hessian matrix will retain its positive definiteness. This condition can be imposed by using a line search method which satisfies the following relation:

$$g_{k+1}^T \Delta x_k \geq \sigma g_k^T \Delta x_k \tag{3.26}$$

where

$$\sigma \in [\tau, 1]$$

and

$$\tau \in [0, \frac{1}{2}]$$

In 1970, Broyden [28], Fletcher [29], Goldfarb [30], and Shanno [31] suggested the BFGS update which is dual with the DFP update. This means that if one applies the DFP method to updating the Hessian matrix from the following:

$$G_{k+1} \Delta x_k = \Delta g_k \tag{3.27}$$

rather than equation (3.11), and then apply the Sherman-Morrison inversion formula to obtain an expression for ΔH_k, the BFGS formula will be obtained:

$$\Delta H_k = \left(1 + \frac{\Delta g_k^T H_k \Delta g_k}{\Delta x_k^T \Delta g_k}\right) \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T \Delta g_k} - \frac{\Delta x_k \Delta g_k^T H_k + H_k \Delta g_k \Delta x_k^T}{\Delta x_k^T \Delta g_k} \tag{3.28}$$

The BFGS update has all the qualities of the DFP method plus it has been noted as working exceptionally well with inexact line searches and a global convergence proof exists [27] for it which is not yet the case for DFP.

3.2.4 Quasi-Newton Updates Through Variational Means

Greenstadt [32] developed a general updating scheme using variational means by minimizing the Euclidean norm of the update. This generated the following updating formula:

$$\Delta H_k = \frac{1}{\Delta g_k^T M \Delta g_k} (\Delta x_k \Delta g_k^T M + M \Delta g_k \Delta x_k^T - H_k \Delta g_k \Delta g_k^T H_k - \frac{1}{\Delta g_k^T M \Delta g_k} (\Delta g_k^T \Delta x_k - (\Delta g_k^T H_k \Delta g_k)) M \Delta g_k \Delta g_k^T M) \tag{3.29}$$

where M is a positive definite matrix. Greenstadt, in his paper, proposed two possibilities for M:

1. $M = H_k$, which results (3.30)
2. $M = I$, which results (3.31)

$$\Delta H_k = \frac{1}{\Delta g_k^T H_k \Delta g_k} (\Delta x_k \Delta g_k^T H_k + H_k \Delta g_k \Delta x_k^T - (1 + \frac{\Delta g_k^T \Delta x_k}{\Delta g_k^T H_k \Delta g_k}) H_k \Delta g_k \Delta g_k^T H_k) \tag{3.30}$$

$$\Delta H_k = \frac{1}{\Delta g_k^T \Delta g_k} (\Delta x_k \Delta g_k^T H_k + H_k \Delta g_k \Delta x_k^T - \Delta g_k \Delta g_k^T H_k - \frac{1}{\Delta g_k^T \Delta g_k} (\Delta g_k^T \Delta x_k - \Delta g_k^T H_k \Delta g_k) \Delta g_k \Delta g_k^T) \tag{3.31}$$

These methods do not retain positive definiteness in general. However, Goldfarb [30] proposed the use of $M = H_{k+1}$ which results in the aforementioned BFGS update and provides a positive definite approximation to the inverse Hessian matrix.

3.3 The Projected Newton Algorithm

Zoutendijk [33] presented a gradient projection method which is summarized by the following steps:

1. Let $P_0 = I$ and start from the initial state x_0.
2. $P_k = I - G_k[G_k^T G_k]^{-1} G_k^T$
   $= P_{k-1} - P_{k-1}\Delta g_k[\Delta g_k^T P_{k-1}\Delta g_k]^{-1}\Delta g_k^T P_{k-1}$
3. If $P_k g_k \neq 0$, let $s_k = -P_k g_k$
4. Minimize $E(x)$ in direction s_k
5. If $P_k g_k = 0$ and $g_k = 0$, then terminate
6. If ($P_k g_k = 0$ and $g_k \neq 0$) or $k = N$, then $k = 0$, $x_0 = x_N$, and go to step 1
7. $k = k + 1$, go to step 2

For a derivation of the projection matrix of step 2 see [34]. The above procedure is equivalent to a quasi-Newton approach with the initial inverse-Hessian approximation $H_0 = I$ which is then updated by

$$H_{k+1} = H_k - \frac{\Delta g_k^T H_k^T H_k \Delta g_k}{\Delta g_k^T H_k \Delta g_k} \tag{3.32}$$

This update is equivalent to the method of Projected Newton-Raphson given by Pearson [26].

All the aforementioned updates belong to a class of updates, Broyden's single parameter family, which is described by the following equations:

$$H_{k+1} = H_k - \frac{H_k \Delta g_k \Delta g_k^T H_k^T}{\Delta g_k^T H_k \Delta g_k} + \theta_k v_k v_k^T + \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T \Delta g_k} \tag{3.33}$$

where

$$v_k = (\Delta g_k^T H_k \Delta g_k)^{\frac{1}{2}} \left(\frac{\Delta x_k}{\Delta x_k^T \Delta g_k} - \frac{H_k \Delta g_k}{\Delta g_k^T H_k \Delta g_k}\right) \tag{3.34}$$

For example, the Broyden family includes the BFGS and DFP updates as special cases. Setting θ_k to 1 in equation (3.33) produces the BFGS update, and setting it to zero gives the DFP update.

4. Initial Scaling of the Inverse Hessian Approximate

Suppose that the objective function E is scaled by a scalar c and this results in a new objective function:

$$E' = cE \tag{4.1}$$

This objective function has the same minimizer as E and its gradient and inverse Hessian are, in terms of those of E,

$$g' = cg \tag{4.2}$$

$$H' = \left(\frac{1}{c}\right) H \tag{4.3}$$

The Newton update for finding the minimizer of a quadratic function, x^*, is $x^* = x_k - Hg_k$. Similarly, the Newton update for E' is

$$x^* = x_k - H' g_k' = x_k - \left(\frac{1}{c}\right) Hcg_k = x_k - Hg_k$$

Therefore, the Newton update is invariant under scaling while quasi-Newton methods are generally not invariant under scaling.

The Broyden family of updates is generally not invariant under scaling. This motivated Oren and Spedicato [35, 36] to modify this family by introducing a new parameter μ_k such that by appropriately choosing μ_k and θ_k, an update which is invariant under scaling of object function can be obtained. This update is given by the following equation:

$$H_{k+1} = \mu_k \left(H_k - \frac{H_k \Delta g_k \Delta g_k^T H_k}{\Delta g_k^T H_k \Delta g_k} + \theta_k v_k v_k^T\right) + \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T \Delta g_k} \tag{4.4}$$

where v_k is given by (3.34b).

Equation (4.4), when used with exact line searches, will become a member of Huang's family [37] where, in his notation,

$$\rho_k = \frac{1}{(\prod_{i=0}^{k} \mu_i)} \tag{4.5}$$

Equation (4.5) leaves a lot of freedom for choosing the parameters μ_k and θ_k such that invariance under scaling may be achieved.

In [38], Spedicato proposed the initialization of H_0 to be the inverse of a diagonal matrix whose diagonal is the diagonal of the true Hessian at x_0. This, however, is not practical for our application since it is very hard to evaluate the Hessian of a multi-layer neural network. Shanno and Phua [13] proposed an initial scaling such that

$$H_0 = \lambda_0^* H_0' \tag{4.6}$$

where λ_0^* is the initial step size given by the line search algorithm, and H_0' is the initial guess for the inverse Hessian. This makes Broyden's one parameter class of updates, given by equation (3.33), self-scaling and invariant under scaling of the objective function.

Another initial scaling, proposed by Shanno and Phua [13], uses Oren-Spedicato's SSVM algorithm and finds the μ_0 provided by that algorithm which minimizes the condition number of $(H_k^{-1} H_{k+1})$ [14, 34–35]. This choice will put a bound on the condition number of the inverse Hessian approximate and, therefore, will provide numerical stability. Minimizing this condition number, one finds the following relationship between μ_k and θ_k:

$$\theta_k = \frac{b(c - b\mu_k)}{\mu_k(ac - b^2)} \tag{4.7a}$$

where

$$a = \Delta g_k^T H_k \Delta g_k \tag{4.7b}$$

$$b = \Delta x_k^T \Delta g_k \tag{4.7c}$$

and

$$c = \Delta x_k^T H^{-1} \Delta x_k \tag{4.7d}$$
$$= \lambda_k^{*2} g_k^T H_k g_k$$

Then it scales the initial estimate of the inverse Hessian by that value. For example, consider the BFGS update for which $\theta_0 = 1$. μ_0 is given by equation (4.7a) to be

$$\mu_0 = \frac{b}{a} \tag{4.8}$$

Since μ should be equal to one for the BFGS method, the initial estimate of H is scaled by μ_0:

$$H_0 = \frac{b}{a} H_0' \tag{4.9}$$

This initial scaling can be evaluated in the same manner for all the members of the Broyden single parameter class of updates using (4.7a) and the appropriate θ_0 for that update.

These initial scalings were shown by Shanno and Phua [13] to improve the performance of the BFGS method over the SSVM methods of Oren and Spedicato in all the cases tested but the case of homogeneous objective functions.

5. Simulations and Results

Feedforward neural network learning algorithms based on most of the aforementioned classical and initial-scaling quasi-Newton methods are evaluated by computer simulations. In all simulations, inexact line searches have been used unless otherwise indicated. Inexact line searches, in general, require fewer function evaluations if it is appropriate for a learning method. In the tables of this paper, E is the final value of the objective function, P is the number of presentations of the set of patterns which is equivalent to the number of evaluations of the objective function, and G is the number of new directions generated by the scheme which for quasi-Newton methods is equivalent to the number of updates made to the inverse Hessian estimate and the number of gradient evaluations. Finally, F is the number of floating point operations. In these tables, for the cases in which local minima were reached, only E is given. A list of the abbreviations used in the tables is given at the beginning of this article.

5.1 A Comparison of Learning Methods

The learning algorithms are used for finding weights and thresholds of two neural networks to emulate the logic of an XOR gate and an encoder respectively. The XOR problem is illustrated with Figure 2 which includes the architecture of the net and the desired input-output patterns. In addition, one of the two sets of initial weights and thresholds used in the simulation is also shown. Another set of initial conditions differs from the first set by that the magnitude of initial weights is 5 instead of 1 and the thresholds are ± 5 instead of 0. These two sets of initial conditions are later referred to as XOR I and XOR II.

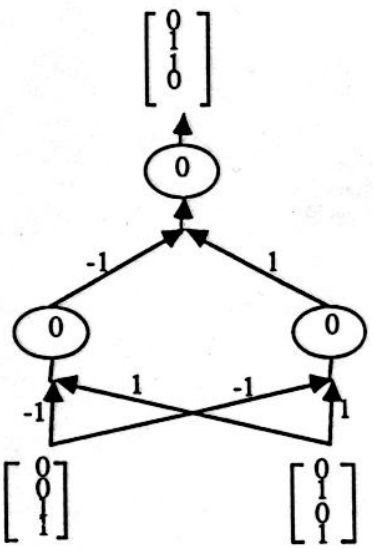

Figure 2. The XOR problem.

The architecture of the net and the desired input-output patterns for the encoder are given in Figure 3 and Table 1. Two important measures of the performance are the number of times the patterns have to be presented and the total number of floating point operations necessary to achieve a near-optimal convergence.

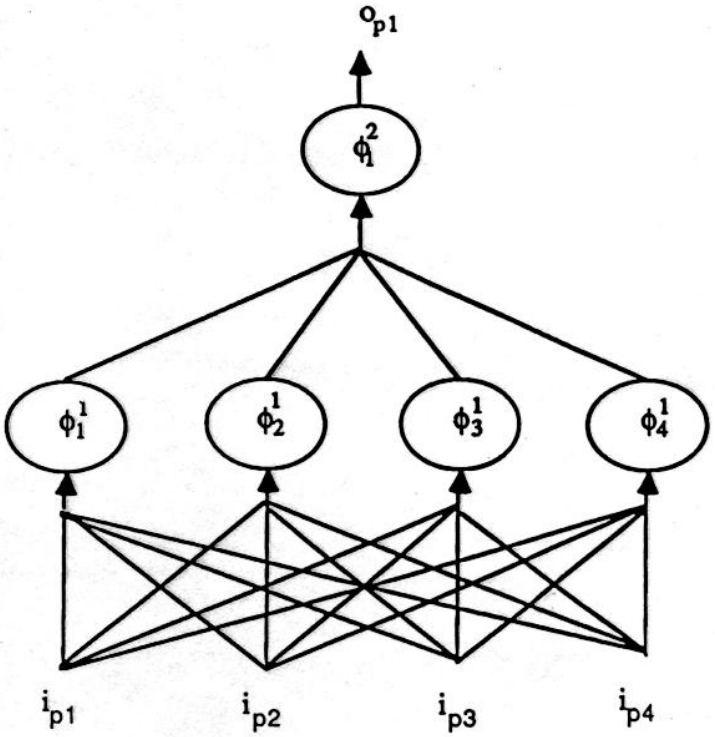

Figure 3. Neural network simulating the encoder logic.

Table 1.
Encoder Logic

I_1	I_2	I_3	I_4	Desired Output
1	1	1	1	0
1	0	1	1	1
1	0	1	0	0
0	0	1	0	1
0	0	0	0	0

5.1.1 Classical Quasi-Newton Methods

The results of the classical Broyden family quasi-Newton methods are given in Table 2. Termination of an algorithm takes place when $E < 1e-4$ or when $|E_{k+1} - E_k| < 1e-5$. The initial conditions in the XOR I problem is possibly close to a ridge of the objective function. This renders the Steepest Descent technique very inefficient. On the other hand, most quasi-Newton methods tested show a better performance, up to two orders of magnitude faster. Pearson II and BFGS find the global minimum with the least number of iterations. Steepest Descent converges an order of magnitude faster for XOR II when compared to XOR I. However, the quasi-Newton methods still have a better performance. Again, Pearson II and BFGS converge to a near global minimum. However, the DFP, Broyden and projected Newton-Raphson (Zoutendijk) methods also converge to the near global minimum at much the same speed as BFGS and Pearson II. Similar performance improvement for quasi-Newton methods is also observed in the encoder problem with Pearson II and BFGS as the best performers. Also, it is clear that quasi-Newton methods converge to points that are much closer to the global minimum ($E = 0$). Namely, mroe precise neural networks are obtained.

Table 2.
Results of Classical Quasi-Newton Methods

		Broyden	DFP	BFGS	PR II	PR III	PNR	GR I	GR II	SD
XOR I	E	0.956	0.666	4 e-16	5.7e-7	0.989	0.955	0.989	0.989	2.8 e-5
	P	—	—	15	36	—	—	—	—	8553
	G	—	—	6	11	—	—	—	—	8465
	F	—	—	1.5 e4	2.2 e4	—	—	—	—	8 e6
XOR II	E	8 e-6	3.5e -6	9.6 e-6	2 e-4	0.952	0.014	0.226	0.226	7 e-6
	P	84	58	53	108	—	31	—	—	163
	G	20	17	15	24	—	8	—	—	52
	F	6 e4	7.7 e4	4.3 e4	6.5 e4	—	6.5 e4	—	—	1 e6
Encoder	E	7.8 e-7	3 e-7	6 e-7	1.9 e-7	1.092	0.125	1.098	0.9611	8 e-5
	P	34	21	17	30	—	—	—	—	1352
	G	7	9	7	10	—	—	—	—	1194
	F	7 e4	1 e5	8.7 e4	8.9 e4	—	—	—	—	3.8 e6

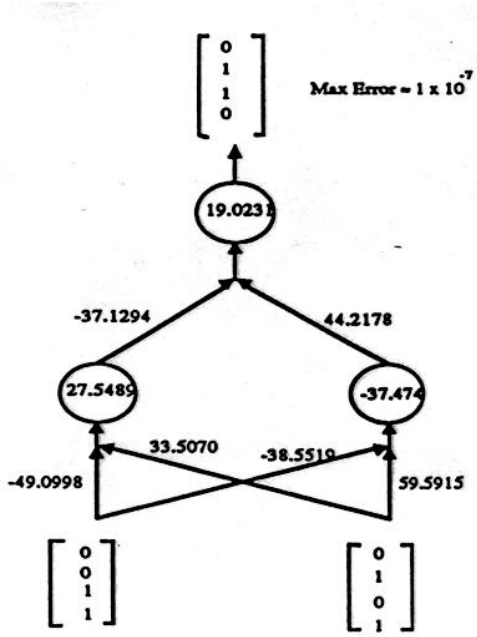

Figure 4. Final state of the neural network simulating the XOR logic as produced by the BFGS method.

5.1.2 Initial-Scaling for Quasi-Newton Methods

Learning based on quasi-Newton methods with initial scaling methods described by (4.6) and (4.9) are also evaluated with the XOR gate and the encoder emulation problems. Results of the computer simulations are given in Table 3 together with those obtained for the BFGS algorith. BFGSa and BFGSb in Table 3 correspond to BFGS with initial scalings by (4.6) and (4.9), respectively.

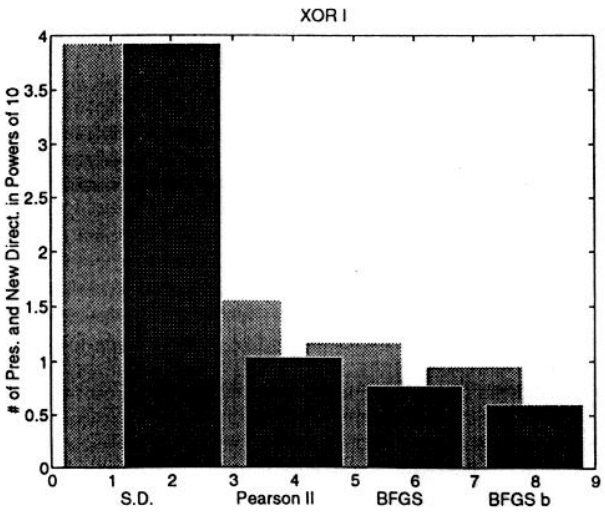

Figure 5. Evaluation of quasi-Newton methods for the XOR I problem.

Table 3.
Results of BFGS Method with Initial Scalings a and b

		BFGS	BFGSa	BFGSb
XOR I	E	4 e-16	7.27 e-5	0
	P	15	14	9
	G	6	6	4
	F	1.5 e4	1.5 e4	1 e4
XOR II	E	9.6 e-6	8.8 e-5	3.7 e-5
	P	53	37	49
	G	15	10	13
	F	4.3 e4	3.5 e4	4 e4
Encoder	E	6 e-7	3 e-7	2 e-5
	P	17	19	23
	G	7	7	9
	F	8.7 e4	8.9 e4	1.1 e5

5.2 Application to Handwriting Recognition

To demonstrate the new learning algorithms, an application of online handwriting recognition was developed which captures the $< x, y >$ coordinates of a stylus pen moving on a digitizer tablet when writing the digits 0 through 9 three times. The sampling rate of the tablet which was used is 90 Hertz and its spatial resolution is 254 dots per inch. The online data is size-normalized, and it is broken up into elementary pieces with the boundaries of these pieces located at the zero y-velocity points in the data. Figure 7 shows the data that was used in this experiment. Figure 8 shows the average number of segments each character was broken into. These segments are called "frames." For each frame of the data, there is a six-dimensional parameter vector of physical features which were found based on a dynamic model of the generation of handwriting. Each frame is also assigned a vector that shows which character the frame was generated from. This vector in the training is a vector of ten values for which the entry corresponding to the character of origin is set to 1 and all other nine entries are set to 0. The two vectors correspond to the input and desired outputs of the network, respectively. Therefore, the network has six inputs and ten outputs (each of which corresponds to one digit).

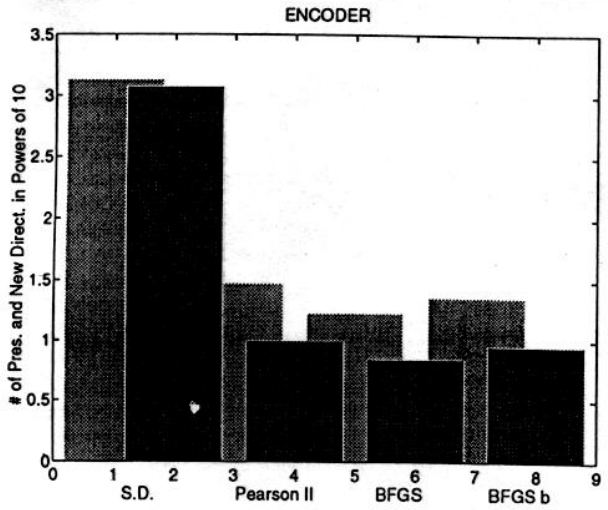

Figure 6. Evaluation of the quasi-Newton methods for the encoder problem.

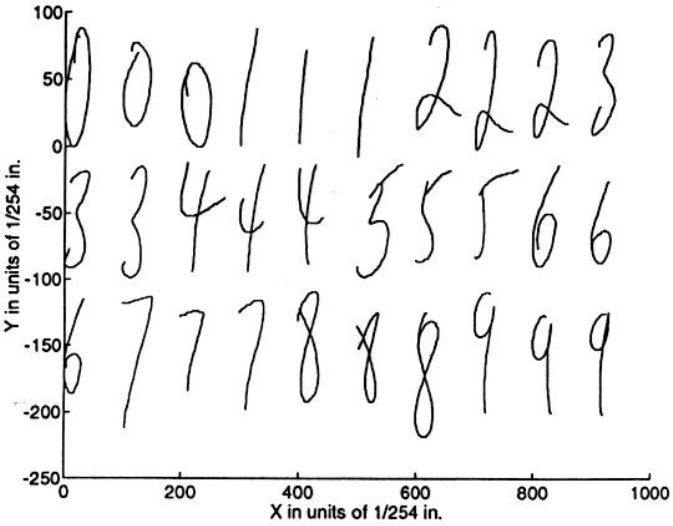

Figure 7. Digits used in the handwriting recognition application.

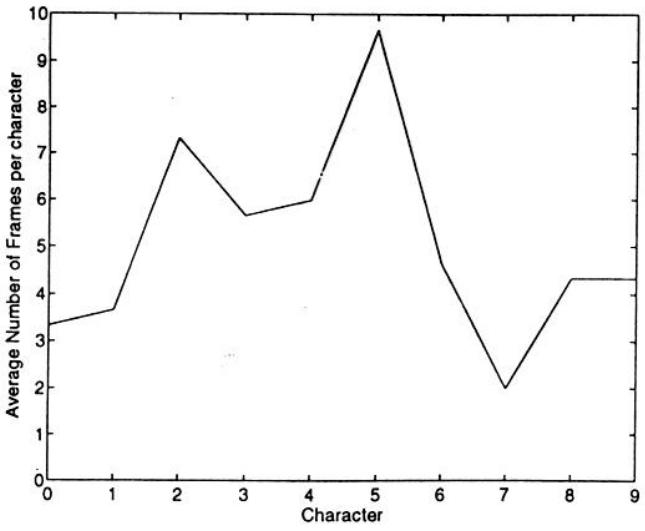

Figure 8. Average number of frames per character.

The architecture used in this application had eleven hidden units and the initial weights were alternately set to 1 and -1. Using BFGS with initial scaling "b," a total of 561 Hessian updates and 1,610 presentations of the 153 frames from the 30 samples of writing, led to a trained system with sum of squares of errors at the output layer of 67.5. This sum of squares of errors is computed over all characters and all 153 presentations. This translates to an accuracy of 96.66% which means that out of 30 samples of digits (3 of each digit) 29 were correctly classified after training. The only mistake was the misrecognition of a character "7" as "4." These results are presented in Figure 9 with the misrecognized character marked as such. In the figure, the characters are sorted such that every three consecutive characters are different samples of some digit going from 0 to 9.

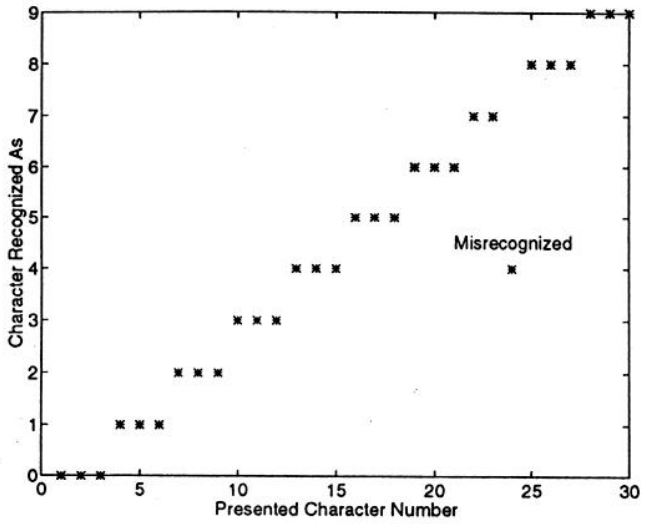

Figure 9. Outcome of the recognition.

6. Conclusion

The need for faster learning algorithms and more accurate networks for applying neural networks to real problems is obvious. It is well understood that for the clss of neural network objective functions, steepest descent techniques are known to perform better away from the minima, and Newton's method works better in the vicinity of the minima. Therefore, to obtain the optimal results, one should take advantage of the best of the two. However, evaluation and inversion of the Hessian matrix which is needed by Newton's method poses a difficult and time-consuming problem for neural network learning.

Classical quasi-Newton methods start off with an estimate of the inverse Hessian matrix equal to the identity matrix which provides the steepest descent direction in the update. As iterations go on, the quasi-Newton methods provide an update to the inverse Hessian matrix. This will, in the limit, provide an optimal direction of descent based on the momentum of the inverse Hessian amtrix.

Results show an increase in the convergence rate of about two to three orders of magnitude by quasi-Newton

methods when compared to the steepest descent method. Pearson II and BFGS methods showed the best performance in all the classical quasi-Newton methods tested. Other quasi-Newton methods converged to local minima which does not show any weakness on their part. The problem with local minima is faced by all learning algorithms for neural networks whose error functions contain many local minima. We also found that condition numbers of the inverse Hessian approximates given by the BFGS method were on average lower than those of the other classical methods such as DFP, etc. This explains, in part, the better convergence rate of the BFGS algorithm.

The Pearson II algorithm is a much simpler algorithm than BFGS and, therefore, uses less number of floating point oeprations (though more iterations) to converge. However, Pearson II does not guarantee a positive definite inverse Hessian approximation while BFGS does, provided the right line search is used (Sec. 3.2.3). Therefore, there is a trade-off between the number of floating point operations and the confidence on positive definiteness of the inverse Hessian approximation (descent in direction).

Quasi-Newton methods with initial scaling of the approximate inverse Hessian matrix are known to perform well especially for problems with a large number of variables [13]. This fact is illustrated by the outstanding performance of the BFGSa in XOR II and BFGSb in XOR I compared to BFGS without initial scaling. However, no major change is seen in the application of the initial scalings to the encoder problem. This is due to the fact that the initial scaling factor comes out very close to "one," meaning that the identity matrix is the best choice for the initial guess of the inverse Hessian matrix. Optimal conditioning introduced by the initial scaling is also an important reason for the acceleration of convergence of the BFGS algorithm.

Figures 5 and 6 show a comparison of the number of presentations and the number of FLOPs required for convergence among steepest descent, Pearson II, BFGS, and BFGS with initial scaling algorithms for the XOR I and encoder problems. Note that the charts are provided on a logarithmic scale and show two to three orders of magnitude reduction in the number of presentations and FLOPs from the steepest descent technique to the quasi-Newton methods.

For comparison purposes, large problems cannot be used due to the fact that neural networks possess many local minima. The number of these local minima grows with the complexity of the network (number of hidden units). Therefore, the XOR and Encoder problems were used for this purpose. However, to demonstrate that the best of the methods presented here will not break down under application to larger problems, the online handwriting recognition problem for digits was formulated and tested. Of course, for a real handwriting recognizer, lots of data is needed and a good test of the accuracy of the system would have to be established through training and testing on different data sets. However, in this paper, since the main point was not to present the robustness of the physical features, the clustering and labeling capabilities of the new learning method were examined through using the same data for training as testing. Having used different data sets for this purpose would not show the clustering capabilities of the network in such a clear way as was done here. The claim here is that, given a set of feature vectors, the learning algorithm converges quickly, generating a set of weights which do a good job of labeling the data (96.66% in this case). For more detailed explanations of handwriting recognition techniques, see references [39] and [40].

References

[1] D.E. Rumelhart, G.E. Hinton, & R.J. Williams, "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 (MIT Press, 1986), 675–695.

[2] H.S.M. Beigi & C.J. Li, "A New Set of Learning Algorithms for Neural Networks." *ISMM Int. Symp. on Computer Applications in Design, Simulation and Analysis*, New Orleans, LA, March 1990, 277–280.

[3] H.S.M. Beigi & C.J. Li, "Neural Network Learning Based on Quasi-Newton Methods with Initial Scaling of Inverse Hessian Approximate." *The 1990 IEEE Long Island Student Conference on Neural Networks*, Old Westbury, NY, April 21, 1990, 49–52.

[4] R.P. Lippmann, "An Introduction to Computing with Neural Nets." *IEEE ASSP Magazine*, April 1987, 4–22.

[5] D.B. Parker, "A Comparison of Algorithms for Neuron-Like Cells." *Neural Networks for Computing*, AIP Conf. Proc. 151, Snowbird, Utah, 1986, 327–332.

[6] S.E. Fahlman, "Faster-Learning Variations on Back-Propagation: An Empirical Study." 1988 Connectionist Models Summer School, 1988, 38–51.

[7] T.P. Vogl, J.K. Mangis, A.K. Rigler, W.T. Zink, & D.L. Alkon, "Accelerating the Convergence of the Back-Propagation Method." *Biological Cybernetics, 59*, 1988, 257–263.

[8] S.A. Solla, E. Levin, & M. Fleisher, "Accelerated Learning in Layered Neural Networks." *Complex Systems, 2*, 1988, 625–640.

[9] A.J. Owens & D.L. Filkin, "Efficient Training of the Back Propagation Network by Solving a System of Stiff Ordinary Differential Equations." Submitted for acceptance at IEEE/INNS Int. Conf. on Neural Networks, Washington, DC, June 19–22, 1989.

[10] H.S.M. Beigi & C.J. Li, "New Neural Network Learning Based on Gradient-Free Optimization Methods." 1990 IEEE Long Island Student Conf. on Neural Networks, Old Westbury, NY, April 21, 1990, 9–12.

[11] D.M. Himmelblau, *Applied Nonlinear Programming* (New York: McGraw-Hill Book Co., 1972).

[12] S. becker & Y. Le Cunn, "Improving the Convergence of Back-Propagation Learning with Second Order Methods." 1988 Connectionist Models Summer School, 1988, 29–37.

[13] D.F. Shanno & K.-H. Phua, "Matrix Conditioning and Nonlinear Optimization." *Mathematical Programming, 14*, 1978, 149–160.

[14] H.S.M. Beigi & C.J. Li, "Learning Algorithms for Neural Networks based on Quasi-Newton Methods with Self-Scaling." *J. Dynamic Systems, Measurement and Control,* March 1993 (Also in *Proc. of Intelligent Control Systems, DSC-23,* Winter Annual Meeting of the ASME, Dallas, TX, Nov. 25–30, 1990, 23–28).

[15] J. Greenstadt, "On the Relative Effectiveness of Gradient Methods." *Mathematics of Computation, 21,* 1967, 360–367.

[16] D.W. Marquardt, "An Algorithm of Least Squares Estimation of Nonlinear Parameters." *SIAM Journal, 11,* 1963, 431.

[17] K.L. Levenberg, *Quart. Appl. Math., 2,* 1944, 164.

[18] S.M. Goldfeld, R.E. Quandt, & H.F. Trotter, "Maximization by Quadratic Hill Climbing." *Econometrica, 34,* 1966, 541.

[19] W.C. Davidon, "Variable Metric Method for Minimization." *AEC Res. and Dev. Report, ANL-5990,* 1959.

[20] J.G.P. Barnes, "An Algorithm for Solving Nonlinear Equations Based on the Secant Method." *Computer Journal, 8,* 1965, 66–72.

[21] C.G. Broyden, "A Class of Methods for Solving Nonlinear Simultaneous Equations." *Maths. Comp., 19,* 1965, 577–593.

[22] C.G. Broyden, "Quasi-Newton.Methods and Their Application to Function Minimization." *Maths. Comp., 21,* 1967, 368–381.

[23] W.C. Davidon, "Variance Algorithms for Minimization." *Computer Journal, 10,* 1968, 406–410.

[24] A.V. Fiacco & G.P. McCormick, *Nonlinear Programming* (New York: John Wiley, 1968).

[25] B.A. Murtagh & E.W.H. Sargent, "A Constrained Minimization Method with Quadratic Convergences." In *Optimization,* R. Fletcher (Ed.) (London: Academic Press, 1969), Ch. 14.

[26] J.D. Pearson, "Variable Metric Methods of Minimization." *Computer Journal, 12,* 1969, 171–178.

[27] R. Fletcher, *Practical Methods of Optimization* (New York: John Wiley & Sons, 1987), 54–56.

[28] C.G. Broyden, "The Convergence of a Class of Double Rank Minimization Algorithms." Parts I and II. *J. Inst. Maths. Applns., 6,* 1970, 76–90, 222–231.

[29] R. Fletcher, "A New Approach to Variable Metric Algorithms." *Computer Journal, 8,* 1970, 317–322.

[30] D. Goldfarb, "A Family of Variable Metric Methods Derived by Variational Means," *Maths. Comp., 24,* 1970, 23–26.

[31] D.F. Shanno, "Conditioning of Quasi-Newton Method for Function Minimization." *Maths. Comp., 24,* 1970, 647–656.

[32] J. Greenstadt, "Variations on Variable-Metric Methods." *Maths. Comp., 24,* 1970, 1–22.

[33] G. Zoutendijk, *Methods of Feasible Directions* (New York: American Elsevire Publishing Company, 1960).

[34] A.S. Householder, *The Theory of Matrices in Numerical Analysis* (Mass.: Blaisdell Publishing Company, 1964), 81.

[35] S.S. Oren, "On the Selection of Parameters in Self Scaling Variable Metric Algorithms." *Mathematical Programming, 7,* 1974, 351–367.

[36] S.S. Oren & E. Spedicato, "Optimal Conditioning of Self-Scaling Varaible Metric Algorithms." *Mathematical Programming, 10,* 1976, 70–90.

[37] H.Y. Huang, "Unified Approach to Quadratically Convergent Algorithms for Function Minimization." *J. of Optimization Theory and Applications, 5*(6), 405–422.

[38] E. Spedicato, "Computational Experience with Quasi-Newton Algorithm for Minimization Problems of Moderately Large Size." *report CISE-N-175, CISE Documentation Service,* Segrate (Milano), 1975.

[39] T. Fujisaki, H.S.M. Beigi, C.C. Tappert, M. Ukelson, & C.G. Wolf, "On-line Recognition of Unconstrained Handprinting: A Stroke-based System and Its Evaluation." In S. Impedovo & J.C. Simon (Eds.), *Pixels to Features III: Frontiers in Handwriting Recognition* (1992: Elsevier Science Publishers, B.V.), 297–312.

[40] T. Fujisaki, K. Nathan, W. Cho, & H. Beigi, "On-line Unconstrained Handwriting Recognition by a Probabilistic Methods." *3rd Int. Workshop on Frontiers of Handwriting Recognition,* Buffalo, New York, May 25–27, 1993.

Biographies

Homayoon S.M. Beigi was born in Tehran, Iran in 1964. He received his B.S., M.S., and D.Eng.Sc. degrees from the Department of Mechanical Engineering at Columbia University in 1984, 1985, and 1990 respectively. His Doctoral thesis was on Learning Control and Neural Network Learning. He worked on lossless image compression at the Center for Telecommunication Research at Columbia University for six months after receiving his Ph.D. degree and joined the handwriting recognition group at IBM T.J. Watson Research Center in New York in February 1991, where he is currently a Research Staff Member. Dr. Beigi has been the recipient of two best paper awards from the Institute of Electrical and Electronic Engineers for his work in Neural Network Learning. He is Associate Editor of the *Intelligent Automation and Soft Computing Journal*, the Editor for the handwriting recognition chapter of the *Berkeley Initiative in Soft Computing*, and Executive Committee Member of the Society for Technological Advancement in Developing Countries. At IBM Research, he has been working on all aspects of handwriting recognition researach such as architecture, preprocessing and front-end, classification, search, language models, training, etc. He is also actively conducting research in Control Theory and Neural Network Learning.

C. James Li received his Ph.D. from the University of Wisconsin in 1987. He joined Columbia University where he was later promoted to associate professor in 1993, shortly before he joined RPI faculty. His areas of research are mechanical diagnostics, and control of manufacturing equipment/processes in which he has published more than 40 papers. Dr. Li has been the principal investigator on contracts and grants from governmental agencies including NSF, ONR, NYSERDA, NSWC and from companies such as IBM, AT&T, and SME. He is a member of ASME where he also serves as a Dynamic System and Control liaison of the Production Engineering Division.