LEARNING ALGORITHMS FOR NEURAL NETWORKS BASED ON QUASI-NEWTON METHODS WITH SELF-SCALING

Homayoon S. M. Beigi and C. James Li
Department of Mechanical Engineering
Columbia University
New York, New York

ABSTRACT

This paper describes a set of new learning algorithms based on a class of Quasi-Newton optimization techniques called Self-Scaling Variable Metric (SSVM) methods. Simulations using the SSVM algorithms for the learning of general feedforward neural networks were done to study their performance compared to other learning algorithms. In particular, the results are compared to those of Quasi-Newton based learning algorithms which were demonstrated to be one to two orders of magnitude faster than the backward propogation.[1] Pearson II and BFGS were reported[1] to be the best of the Quasi-Newton methods investigated. However, SSVM methods were shown to be 100% better than the Pearson II and BFGS.

INTRODUCTION

The simplest neuron in a neural network sums a number of weighted inputs and passes the result through a non-linear activation function o. Multi-layer neural networks consist of a large number of these neurons. Before a neural network can be used for any purpose, the weights connecting inputs to neurons and parameters of the activation functions should be adjusted so that outputs of the network will match desired patterns for specific sets of inputs. The methods used for adjusting these weights and parameters to provide such a match are usually refered to as learning algorithms.

Rumelhart et al.[2] introduced the Back-Propagation learning algorithm for multi-layer neural nets in 1986. They used the generalized delta rule to compute the needed gradient for this steepest decent method. However, low rates of convergence were seen in practically every problem. Lippmann[3] states, "One difficulty noted by the backward 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.)" A few methods[3-4] have been proposed to increase the rate of convergence of learning by making strong assumptions such as linearity. Other more practical methods have recently been proposed for speeding the convergence of the back-propagation technique.[5-8]

In general, steepest descent techniques have good performance while away from a local minimum and require a lot of iterations to converge while close to the minimum. On the other hand, Newton's method usually converges fast in the vicinity of the minimum. In addition, Newton's minimization technique handles functions with ill-conditioned Hessian matrices elegantly [9]. It would be desirable to take advantage of the properties of steepest descent when the state is far from the minimum and then to use Newton's method in the vicinity of the minimum.

For using Newton's method, the first gradient and the matrix of second partial derivatives (Hessian) should be evaluated. The moment one talks about evaluating the Hessian matrix, it becomes clear that a layer by layer adjustment of the weights is not possible and there are elements of the Hessian which are related to elements in different layers. Due to the complexity faced in evaluating the Hessian matrix, one would settle for its approximation. One way would be to look at the problem in a manner similar to Rumelhardt's technique and to use a momentum method which would approximate the diagonal elements of the Hessian matrix and stays ignorant of the off-diagonal elements.[10]

The difficulties associated with the use of Newton's method for neural network learning are: 1) the complexity of the evaluation of the Hessian, 2) the inversion of the Hessian, and 3) the opitmal switching from steepest descent to Newton's method.

Quasi-Newton methods provide a solution to all three difficulties by providing an iterative estimate for the inverse of the Hessian matrix. If one selects the initial estimate to be an identity matrix, it is a steepest descent technique at the beginning and gradually changes into Newton's method as the estimate becomes the inverse of the Hessian. In this paper, a general formulation is provided for the evaluation of the gradients and objective function of minimization in multi-layered feed-forward neural networks. In [1], the authors investigated nine different Quasi-Newton methods and the results were compared to a steepest descent technique. This paper looks at a different set of Quasi-Newton methods called Self-Scaling Variable Metric methods which provide directions that are invariant under scaling of the objective function.[11-12]

PROBLEM FORMULATION

State Vector and Gradient Evaluation

The objective is to minimize the output error of the top (output) layer (layer L) of a neural network over a set of 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 number)
ω^l_{nm}	(Weighting factor between the m^{th} input and neuron n in layer l)
o^l_{pn}	(Output of neuron n of 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,

$$\text{minimize } E = \sum_{p=1}^{P} \sum_{n_L=1}^{N_L} (o^{L}_{pn_L} - t_{pn_L})^2 \qquad (1)$$

Define,

$$E_p = \sum_{n_L=1}^{N_L} (o^{L}_{pn_L} - t_{pn_L})^2$$

then,

$$E = \sum_{p=1}^{P} E_p \qquad (2)$$

Let us define a few variables and eventually a state vector which would include all the variables to be optimized for minimum E:

$$\phi^{lT} = [\, \phi^{l}_1, \phi^{l}_2, \ldots, \phi^{l}_{N_l} \,]$$

(Activation Function Parameter Vector for Level l)

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

(Vector of intercellular weights to neuron n_l at layer l)

$$\omega^{lT} = [\, \omega^{lT}_1, \omega^{lT}_2, \ldots, \omega^{lT}_{N_l} \,]$$

(Supervector of intercellular weights of level l)

$$x^{lT} = [\, \phi^{lT}, \omega^{lT} \,] \quad \text{(State vector for level } l\text{)}$$

$$x^{T} = [\, x^{1T}, x^{2T}, \ldots, x^{LT} \,] \quad \text{(Super state vector)}$$

Let j denote any member of the state vector x, then,

$$\frac{\partial E}{\partial x_j} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial x_j} \qquad (3)$$

and by chain rule,

$$\frac{\partial E_p}{\partial x_j} = 2 \sum_{n_L=1}^{N_L} (o^{L}_{pn_L} - t_{pn_L}) \frac{\partial o^{L}_{pn_L}}{\partial x_j} \qquad (4)$$

where,

$$\frac{\partial o^{L}_{pn_L}}{\partial \phi^{l}_{n_l}} = \frac{\partial o^{L}_{pn_L}}{\partial o^{l}_{pn_l}} \frac{\partial o^{l}_{pn_l}}{\partial \phi^{l}_{n_l}} \qquad (5)$$

and using index notation,

$$\frac{\partial o^{L}_{pn_L}}{\partial o^{l}_{pn_l}} = \prod_{i=l+1}^{L} \frac{\partial o^{(L+l-i+1)}_{pn_{(L+l-i+1)}}}{\partial o^{(L+l-i)}_{pn_{(L+l-i)}}} \qquad (6)$$

In equation (6) the index notation has been employed, i.e.,

$$\frac{\partial o^{l+1}_{pn_{l+1}}}{\partial o^{l}_{pn_l}} \frac{\partial o^{l}_{pn_l}}{\partial o^{l-1}_{pn_{l-1}}} \longleftrightarrow \sum_{n_l=1}^{N_l} \frac{\partial o^{l+1}_{pn_{l+1}}}{\partial o^{l}_{pn_l}} \frac{\partial o^{l}_{pn_l}}{\partial o^{l-1}_{pn_{l-1}}} \text{ (traditional notation)}$$

Take, for example, the logistic function for the activation function o,

$$o^{l}_{pn_l} = \frac{1}{1 + e^{-s^{l}_{pn_l}}} \qquad (7)$$

where,

$$s^{l}_{pn_l} = \sum_{n_{l-1}=1}^{N_{l-1}} (o^{l-1}_{pn_{l-1}} \omega^{l}_{n_l n_{l-1}}) + \phi^{l}_{n_l}$$

then,

$$\frac{\partial o^{l}_{pn_l}}{\partial \phi^{l}_{n_l}} = d^{l}_{pn_l} \qquad (8)$$

$$\frac{\partial o^{l}_{pn_l}}{\partial \omega^{l}_{n_l n_{l-1}}} = d^{l}_{pn_l} o^{l-1}_{pn_{l-1}} \qquad (9)$$

and,

$$\frac{\partial o^{l}_{pn_l}}{\partial o^{l-1}_{pn_{l-1}}} = d^{l}_{pn_l} \omega^{l}_{n_l n_{l-1}} \qquad (10)$$

where,

$$d^{l}_{pn_l} = \frac{e^{-s^{l}_{pn_l}}}{(1 + e^{-s^{l}_{pn_l}})^2} \qquad (11)$$

However,

$$o^{l}_{pn_l} = \frac{1}{1 + e^{-s^{l}_{pn_l}}}$$

from which,

$$e^{-s^{l}_{pn_l}} = \frac{1}{o^{l}_{pn_l}} - 1 \qquad (12)$$

Rewrite equation (11) using (12),

$$d^{l}_{pn_l} = o^{l}_{pn_l} (1 - o^{l}_{pn_l}) \qquad (13)$$

From (13), (8), (9), and (10),

$$\frac{\partial o^{l}_{pn_l}}{\partial \phi^{l}_{n_l}} = o^{l}_{pn_l} (1 - o^{l}_{pn_l}) \qquad (14)$$

$$\frac{\partial o^{l}_{pn_l}}{\partial \omega^{l}_{n_l n_{l-1}}} = o^{l-1}_{pn_{l-1}} o^{l}_{pn_l} (1 - o^{l}_{pn_l}) \qquad (15)$$

and,

$$\frac{\partial o^{l}_{pn_l}}{\partial o^{l-1}_{pn_{l-1}}} = \omega^{l}_{n_l n_{l-1}} o^{l}_{pn_l} (1 - o^{l}_{pn_l}) \qquad (16)$$

We can therefore use equations 14-16 and 3-6 to evaluate the elements of gradient of E with respect to the super state vector x.

Example: The Exclusive-OR (XOR) Problem

Take the example of a network of three neurons (figure 1) which is employed to simulate the exclusive-OR (XOR) logic pattern (table 1.)

The objective function of minimization will then be,

$$E = \sum_{p=1}^{4} (o_{p1}^2 - t_{p1})^2$$

and the super state vector is defined by the following sequence of definitions,

$$\phi^{1T} = [\phi_1^1, \phi_2^1]$$

$$\phi^2 = [\phi_1^2]$$

$$\omega^{1T} = [\omega_{11}^1, \omega_{12}^1, \omega_{21}^1, \omega_{22}^1]$$

$$\omega^{2T} = [\omega_{11}^2, \omega_{12}^2]$$

$$x^T = [\phi_1^1, \phi_2^1, \omega_{11}^1, \omega_{12}^1, \omega_{21}^1, \omega_{22}^1, \phi_1^2, \omega_{11}^2, \omega_{12}^2]$$

From equation (4),

$$\frac{\partial E_p}{\partial x_j} = 2 (o_{p1}^2 - t_{p1}) \frac{\partial o_{p1}^2}{\partial x_j}$$

and from equations 14-16,

$$\frac{\partial o_{p1}^2}{\partial \phi_1^2} = o_{p1}^2 (1 - o_{p1}^2)$$

$$\frac{\partial o_{p1}^2}{\partial \omega_{11}^2} = o_{p1}^1 o_{p1}^2 (1 - o_{p1}^2)$$

$$\frac{\partial o_{p1}^2}{\partial \omega_{12}^2} = o_{p2}^1 o_{p1}^2 (1 - o_{p1}^2)$$

$$\frac{\partial o_{p1}^2}{\partial \phi_1^1} = \frac{\partial o_{p1}^2}{\partial o_{p1}^1} \frac{\partial o_{p1}^1}{\partial \phi_1^1} = \omega_{11}^2 o_{p1}^2 (1 - o_{p1}^2) o_{p1}^1 (1 - o_{p1}^1)$$

$$\frac{\partial o_{p1}^2}{\partial \phi_2^1} = \frac{\partial o_{p1}^2}{\partial o_{p2}^1} \frac{\partial o_{p2}^1}{\partial \phi_2^1} = \omega_{12}^2 o_{p1}^2 (1 - o_{p1}^2) o_{p2}^1 (1 - o_{p2}^1)$$

$$\frac{\partial o_{p1}^2}{\partial \omega_{11}^1} = \frac{\partial o_{p1}^2}{\partial o_{p1}^1} \frac{\partial o_{p1}^1}{\partial \omega_{11}^1} = \omega_{11}^2 o_{p1}^2 (1 - o_{p1}^2) i_{p1} o_{p1}^1 (1 - o_{p1}^1)$$

$$\frac{\partial o_{p1}^2}{\partial \omega_{12}^1} = \frac{\partial o_{p1}^2}{\partial o_{p1}^1} \frac{\partial o_{p1}^1}{\partial \omega_{12}^1} = \omega_{11}^2 o_{p1}^2 (1 - o_{p1}^2) i_{p2} o_{p1}^1 (1 - o_{p1}^1)$$

$$\frac{\partial o_{p1}^2}{\partial \omega_{21}^1} = \frac{\partial o_{p1}^2}{\partial o_{p2}^1} \frac{\partial o_{p2}^1}{\partial \omega_{21}^1} = \omega_{12}^2 o_{p1}^2 (1 - o_{p1}^2) i_{p1} o_{p2}^1 (1 - o_{p2}^1)$$

$$\frac{\partial o_{p1}^2}{\partial \omega_{22}^1} = \frac{\partial o_{p1}^2}{\partial o_{p2}^1} \frac{\partial o_{p2}^1}{\partial \omega_{22}^1} = \omega_{12}^2 o_{p1}^2 (1 - o_{p1}^2) i_{p2} o_{p2}^1 (1 - o_{p2}^1)$$

Using the above equations, the elements of $\nabla_x E_p$ and using equation (3) the elements of $\nabla_x E$ are found.

Define,

$$g = \nabla_x E = \sum_{p=1}^{P} \nabla_x E_p \qquad \text{(Gradient Vector)}$$

and,

$$G = \nabla_x^2 E = \sum_{p=1}^{P} \nabla_x^2 E_p \qquad \text{(Hessian Matrix)}$$

Most minimization techniques require gradient evaluations of the objective function. The above formulation is used in most of the following minimization techniques.

Variable Metric (Quasi-Newton) Minimization Techniques

Let x^* denote the minimum of the objective function E. Further assume that the current state is x_k and define Δx_k to be the difference between the states at k and k+1 iterations, namely,

$$x_{k+1} = x_k + \Delta x_k \qquad (17)$$

Write the Taylor series expansion of $E(x^*)$ about x_k assuming that every update of the state vector should drive the state vector to optimal value x^*, namely, $x^* = x_k + \Delta x_k$.

$$E(x^*) = E(x_k) + \nabla_x^T E|_{x_k} \Delta x_k + \Delta x_k^T \nabla_x^2 E|_{x_k} \Delta x_k + O(\Delta x_k^3) \qquad (18)$$

By using the previously defined symbols for the first and second gradients of E (g and G,) with a subscript k to denote their evalation at x_k, we may write (18) as,

$$E(x^*) = E(x_k) + g_k^T \Delta x_k + \Delta x_k^T G_k \Delta x_k + O(\Delta x_k^3) \qquad (19)$$

If we disregard the higher than second order terms in (19) and thus approximate E with a quadratic function in the vicinity of x_k and x^*, then the quadratic approximation of (19) may be written as follows:

$$E(x^*) \approx E(x_k) + g_k^T \Delta x_k + \Delta x_k^T G_k \Delta x_k \qquad (20)$$

Note that for a minimum of $E(x^*)$, a necessary condition is that $\nabla_{x^*} E$ be zero. However, keeping the current state x_k constant and then taking the gradients of both sides of (20), since $x^* = x_k + \Delta x_k$,

$$\nabla_x E \approx g_k + G \Delta x_k \qquad (21)$$

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

$$g_k + G \Delta x_k \approx 0 \qquad (22)$$

or,

$$\Delta x_k \approx - G_k^{-1} g_k \qquad (23)$$

Decompose Δx_k into a direction s_k and a magnitude λ,

$$\Delta x_k = \lambda_k s_k \qquad (24)$$

Then, since E is not a quadratic function in x_k, then the following recursive update could be used for the state vector,

$$x_{k+1} = x_k + \lambda_k^* s_k \qquad (25)$$

where λ_k^* is the optimum step size in the direction s_k, and s_k is given by the following,

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

A line search method could be used to find λ_k^* for direction s_k.

This method provides quadratic convergence and is very powerful in the vicinity of minima. However there are different problems

associated with this approach. The first problem is that in order for E to always descend in value, the matrix G^{-1} should be positive definite. Since E is generally not quadratic, G^{-1} could become indefinite or even negative definite. There are lots of techniques developed to keep a positive definite approximation of the inverse Hessian matrix (G^{-1}) such that the quadratic information in the hessian matrix will be used. Using the quadratic information generally provide a better direction of descent than the steepest descent direction, especially in the vicinity of the minimum. Among these methods for keeping a positive definite approximation of the inverse Hessian matrix are Greenstadt's method[13], Marquardt[14], Levenberg[15], and Goldfeld, Quandt and Tratter's alternative[16].

The second problem is that for networks with a small number of neurons it might be feasible to find the Hessian matrix, however, for larger networks it will become a very difficult task. In addition, it will be very hard to write general equations for the evaluation of the elements of the Hessian matrix as done for the elements of the gradient vector g.

Let us say that a Hessian matrix is calculated at every recursion k. Then a still more serious problem occurs. For huge networks it is not practical to take the inverse of the Hessian matrix. The time involved in taking this inverse in most cases will be more costly than in taking more steps for convergence using a simpler method such as the steepest descent method.

Problem one can be solved by the noted methods above. However, problems two and three make using Newton's method quite impractical. These limitations are reasons for looking at the following alternatives which in turn will solve the discussed problems and still keep a super-linear rate of convergence.

From equation (25), we can write the following generalized recursive algorithm to update the state vector such that a minimum E will be approached:

$$x_{k+1} = x_k - \lambda^*_k H_k \nabla_x E(k) \tag{26}$$

where λ^* is a weighting factor, and H is a square symmetric matrix. Depending on the matrix H_k that is used in equation (26), different optimization algorithms will be provided. Therefore, H_k multiplied by the gradient of E will provide a direction and λ^*_k is the optimal step in that descent direction as provided by some line search method. If H_k in equation (26) is made equivalent to the identity (I) matrix, then the method reduces to the steepest descent technique which provides linear convergence. Making H_k equivalent to the inverse of the Hessian matrix (G^{-1}) as previously defined, the method will reduce to the Newton minimization technique which provides quadratic convergence in the vicinity of the local minima. Quasi-Newton methods will start with an approximation to the inverse-Hessian matrix such as the identity matrix. Variable metric updates for H_k are then used, leading to different types of Quasi-Newton methods. In these methods, a rank one or rank two update is provided for H_k, denoted by ΔH_k, which will lead to the convergence of H to G^{-1} for quadratic functions. Updates to matrix H_k are done recursively in different directions of the inverse-Hessian space, based on the information obtained from the function behavior in that direction. Depending on whether these updates are done in one or two directions at a time, rank one or rank two methods are generated. Quasi-Newton methods in general try to keep the hereditary relation (27a) satisfied.

$$H\,\Delta g_k = \Delta x_k \tag{27a}$$

Condition (27a) is automatically satisfied for a quadratic function if H is the exact inverse Hessian. However, since in Quasi-Newton methods, the inverse Hessian is supposed to be an approximation, instead of $H_k\,\Delta g_k = \Delta x_k$, the methods try to keep the folowing relation satisfied at each step k,

$$H_{k+1}\,\Delta g_k = \Delta x_k \tag{27b}$$

This relationship is refered to as the Quasi-Newton condition and it means that the inverse Hessian matrix should be updated such that relation (27b) is satisfied.

Reference [1] gives a thorough review of some classical rank-one and rannk-two updates used with Quasi-Newton methods. Here, a mere listing of the equations for the Pearson II and BFGS updates is given by equations (28) and (29) respectively.

Pearson's No. 2 update is given by the following:[17]

$$\Delta H_k = \frac{(\Delta x_k - H_k \Delta g_k)\,\Delta x_k^T}{\Delta x_k^T\,\Delta g_k} \tag{28}$$

It should be noted that Pearson's methods do not guarantee retainment of positive definiteness of the inverse Hessian Matrix. Therefore, it is a good idea to reset the inverse Hessian approxiamation to Identity every N iterations. Methods of this nature are called cyclic.

In 1970, Broyden[18], Fletcher[19], Goldfarb[20], and Shanno[21] suggested the BFGS update (29) which is dual with the DFP method.[22]

$$\Delta H_k = \left(1 + \frac{\Delta^T g_k H_k \Delta g_k}{\Delta^T x_k \Delta g_k}\right)\frac{\Delta x_k \Delta x_k^T}{\Delta^T x_k \Delta g_k} - \frac{\Delta x_k \Delta^T g_k H_k + H_k \Delta g_k \Delta x_k^T}{\Delta^T x_k \Delta g_k} \tag{29}$$

The BFGS update has all the qualities of the DFP method plus the fact that it has been noted to work exceptionally well with inexact line searches and a global convergence proof exists[23] for the BFGS. No such proof has been done for the convergence of DFP yet.

Self-Scaling Variable Metric (SSVM) Algorithms

Equation (30) is a Slef-Scaling Variable Metric (SSVM) for the approximation to the inverse Hessian matrix.[11]

$$H_{k+1} = \mu_k \left(H_k - \frac{H_k \Delta g_k \Delta g_k^T H_k}{\Delta g_k^T H_k \Delta g_k} + \theta_k v_k v_k^T\right) + \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T \Delta g_k} \tag{30}$$

where,

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

Note that this update under an exact line search will be a member of Huang's family[23] with,

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

This SSVM algorithm maintains positive definiteness of the approximation to the inverse Hessian matrix, provided $\Delta x_k^T \Delta g_k > 0$ for all k. This condition can be imposed by using a line search method which satisfies the following relation,

$$g_{k+1}^T \Delta x_k \geq \sigma\, g_k^T \Delta x_k \tag{31}$$

where $\sigma \in [\tau, 1]$ and $\tau \in [0, 1/2]$ are parameters of the line search termination.[22] Eventually, this means that the curvature estimate should be positive where the updating is done. The same argument is true for lots of other variable metric methods such as the DFP and BFGS methods.[1] For a general non-linear function, the SSVM algorithm provides a set of search direction which are invariant under scaling of the objective function. Also, for a quadratic objective function, the algorithm has the property that it monotonically reduces the condition number of the inverse Hessian approximate.

Equation 30 leaves a lot of freedom in choosing the parameters μ_k and θ_k. Oren suggested in [11] that μ_k and θ_k be picked in the following manner,

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

and $\phi_k, \theta_k \in [0,1]$. This choice will provide a set of μ_k such that,

$$\frac{\Delta x_k^T \Delta g_k}{\Delta g_k^T H_k \Delta g_k} \le \mu_k \le \frac{\Delta x_k^T H_k^{-1} \Delta x_k}{\Delta x_k^T \Delta g_k}$$

Another approach for picking the aforementioned parameters, based mainly on heuristics, is to keep μ_k as close as possible to unity and to pick θ_k such as to offset an estimated bias in $\det(H_k G)$ relative to unity.[12] A third approach for picking these paramters is to minimize the condition number of the inverse Hessian matrix which will provide better numberical stability of the updating algorithm. Reference [12] gives four sets of switching rules for picking these parameters.

SIMULATIONS AND RESULTS

Computer simulations of the learning have been done using all the described minimization schemes as applied to the classical XOR problem and an encoder. The input and desired output patterns of these two problems are given in tables 1 and 2 respectively. The architecture used for the XOR problem is shown in figure 1. A similar architecture was used for the encoding problem with four hidden neurons and four inputs. This was intended to see the performance of the minimization schemes when there are more variables involved. Two important measures of the performance of the algorithms are the number of times the input patterns had to be presented and the number of updates which were made to the initial guess for the inverse Hessian matrix to achieve convergence. In simulation of all the above algorithms, an inexact line search was used.

In picking the parameters μ_k and θ_k, a trial on error method was used. The switching method of [11] did not perform well compared to pre-chosen constant parameters. This result agrees with experiments done by Oren.[11] (e.g. the number of presentations and updates for the encoder problem were 77 and 30 respectively.) Also, the switching methods of [12] which were demonstrated to have done well were not used here due to their impracticality for neural networks. In these switching methods, an update of the Hessian matrix should be kept as well as the inverse Hessian. This would require a lot of memory when a large network is involved.

Results of the computer simulation of the said algorithms are given in table 3. In table 3, E is the final value of the objective function, P is the number of presentations of all input patterns, and G is the number of updates made to the initial inverse Hessian estimate. Termination of the minimization took place when $E < 1e\text{-}5$ or when $|E_{k+1} - E_k| < 1e\text{-}6$.

CONCLUSION

Looking at the state of the arts in neural network learning, a need was noted for faster learning algorithms. Steepest Descent techniques are known to perform well, away from the minima and Newton's method works well, in the vicinity of the minima. Therefore, for a fast learning algorithm, one should take advantage of the best of the two methods. The evaluation and inversion of the Hessian matrix posed a difficult and time consuming problem. Quasi-Newton methods start off with an estimate of the inverse Hessian matrix equal to the identity matrix. This matrix provides the steepest descent direction. As recursions are done, 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 matrix.

Results show an increase in the rate of convergence of up to 100% in the case of SSVM methods over BFGS and Pearson II. SSVM updates are known to perform well especially for problems with a large number of variables.[22] This fact is illustrated by the outstanding performance of the SSVM method compared to BFGS in the encoder problem where the size of the state vector x is 25.

REFERENCES

1. Homayoon S. M. Beigi, C. James Li, "A New Set of Learning Algorithms for Neural Networks," ISMM International Symposium, Computer Applications in Design, Simulation and Analysis, New Orleans, LA, March 1990.

2. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation", in D. E. Rumelhart and J. L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, 1986, pp. 675-695.

3. R. P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, April 1987, pp. 4-22.

4. D. B. Parker, "A Comparison of Algorithms for Neuron-Like Cells," Neural Networks for Computing, AIP conference Proceeding 151, Snowbird, Utah, 1986, pp. 327-332.

5. S. E. Fahlman, "Faster-Learning Variations on Back-Propagation: *An Empirical Study*," 1988 Connectionist Models Summer School, 1988, pp.38-51.

6. T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the Convergence of the Back-Propagation Method," Biological Cybernetics, Vol. 59, 1988, pp. 257-263.

7. S. A. Solla, E. Levin, M. Fleisher, "Accelerated Learning in Layered Neural Networks," Complex Systems, Vol. 2, 1988, pp. 625-640.

8. A. J. Owens and D. L. Filkin, "Efficient Training of the Back Propagation Network by Solving a System of Stiff Ordinary Differential Equations," *submitted for publication at* IEEE/INNS International Conference on Neural Networks, June 19-22, 1989, Washington D.C.

9. David M. Himmelblau, Applied Nonlinear Programming, p.87, McGraw-Hill Book Company, New York, 1972.

10. S. Becker, Y. Le Cunn, "Improving the Convergence of Back-Propagation Learning with Second Order Methods," 1988 Connectionist Models Summer School, 1988, pp. 29-37.

11. S. S. Oren, "On the Selection of Parameters in Self Scaling Variable Metric Algorithms," Mathematical Programming, Vol.7, 1974, pp.351-367.

12. S. S. Oren and E. Spedicato, "Optimal Conditioning of Self-Scaling Variable Metric Algorithms," Mathematical Programming, Vol. 10, 1976, pp.70-90.

13. J. Greenstadt, Mathematics of Computation, Vol. 21, 1967, p. 360.

14. D. W. Marquardt, Journal of SIAM, Vol. 11, 1963, p.431.

15. K. L. Levenberg, Quart. Appl. Math., Vol. 2, 1944, P.164.

16. S. M. Goldfeld, R. E. Quandt, and H. F. Trotter, Econometrica, Vol.34, 1966, p.541.

17. J. D. Pearson, "Variable Metric Methods of Minimization," Computer Journal, Vol. 12, 1969, p. 171-178.

18. C. G. Broyden, "The Convergence of a Class of Double Rank Minimization Algorithms," Parts I and II, J. Inst. Maths. Applns., Vol. 6, 1970, pp. 76-90, 222-231.

19. R. Fletcher, "A New Approach to Variable Metric Algorithms," Computer Journal, Vol. 8, 1970, pp.317-322.

20. D. Goldfarb, "A Family of Variable Metric Methods Derived by Variational Means, Mathematics of Computation, Vol. 24, 1970, pp.23-26.

21. D. F. Shanno, "Conditioning of Quasi-Newton Method for Function Minimization," Mathematics of Computation, Vol. 24, 1970, pp.647-656.

22. R. Fletcher, Practical Methods of Optimization, pp. 54-56, John Wiley & Sons, New York, 1987.

23. H. Y. Huang, "Unified Approach to Quadratically Convergent Algorithms for Function Minimization," Journal of Optimization Theory and Applications, Vol. 5, No. 6, 1970, pp.405-422.

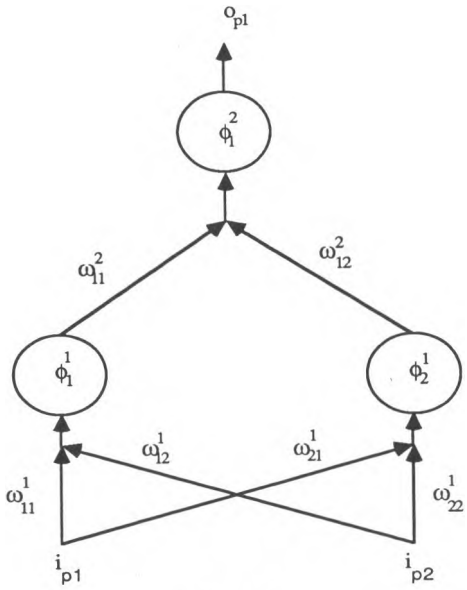

Figure 1

Input Patterns		Output Patterns
0	0	0
1	0	1
0	1	1
1	1	0

Table 1 (XOR)

I1	I2	I3	I4	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

Table 2 (Encoder)

		BFGS	Prsn	O(1,.1)	O(1,.2)	O(1,.3)	O(1,.5)	O(.5,.5)
XOR	E	4e-16	6e-7	9e-9	0	0	----	8e-7
	P	15	36	12	8	11	----	13
	G	6	11	4	3	4	----	5
Encoder	E	1e-17	2e-7	7e-6	8e-6	3.7e-6	2e-11	1e-8
	P	36	54	30	42	33	20	28
	G	11	16	9	13	10	6	9

Table 3

* $O(\phi,\theta)$'s denote the two parameters of equation 30.