Pergamon

Computers chem. Engng Vol. 20, No. 9, pp. 1133–1140, 1996

0098-1354(95)00228-6
0098-1354/96 $15.00 + 0.00

MODIFIED QUASI-NEWTON METHODS FOR TRAINING NEURAL NETWORKS

B. ROBITAILLE,[1] B. MARCOS,[1] † M. VEILLETTE[2] and G. PAYRE[2]

[1]Département de génie chimique and [2]Département de génie mécanique, Université de Sherbrooke, Sherbrooke, Québec J1K 2R1, Canada

(Received 19 July 1993; final revision received 7 August 1995)

Abstract—The backpropagation algorithm is the most popular procedure to train self-learning feedforward neural networks. However, the convergence of this algorithm is slow, it being mainly a steepest descent method. Several researchers have proposed other approaches to improve the convergence: conjugate gradient methods, dynamic modification of learning parameters, quasi-Newton or Newton methods, stochastic methods, etc. Quasi-Newton methods were criticized because they require significant computation time and memory space to perform the update of the Hessian matrix limiting their use to middle-sized problems. This paper proposes three variations of the classical approach of the quasi-Newton method that take into account the structure of the network. By neglecting some second-order interactions, the sizes of the resulting approximated Hessian matrices are not proportional to the square of the total number of weights in the network but depend on the number of neurons of each level. The modified quasi-Newton methods are tested on two examples and are compared to classical approaches like regular quasi-Newton methods, backpropagation and conjugate gradient methods. The numerical results show that one of these approaches, named BFGS-N, represents a clear gain in terms of computational time, on large-scale problems, over the traditional methods without the requirement of large memory space. Copyright © 1996 Elsevier Science Ltd

1. INTRODUCTION

Self-learning feedforward neural nets require a learning process to map an output set to an input set. The learning process determines weights W that characterize the connections between neurons, making the link between inputs and outputs. The backpropagation algorithm as proposed by Rumelhart *et al.* (1986) is the most popular procedure, which involves estimation of a set of weights to get a satisfying relationship between inputs and outputs, as long as high precision is not required. However, this procedure converges slowly, which is not surprising since the backpropagation algorithm is essentially a steepest descent method; in the optimization domain, theoretical and numerical works have shown that the order of convergence of simple gradient methods is at most one, and that the rate of convergence tends to worsen as the size of the problem at hand is increased.

Several accelerating techniques were proposed to speed up the converging procedure. Fahlman (1988), Jacobs (1988) and Tallaneare (1990) proposed to modify dynamically some parameters such as the learning rate or the momentum; Rigler *et al.* (1991) suggested a scaling of the derivatives as a function of successive levels. Leonard and Kramer (1990) improved the backpropagation algorithm with conjugate gradient techniques to adjust dynamically the learning rate and the momentum with a unidirectional search at each optimization step. Van Ooyen and Nienhuis (1992) presented modifications in the error function used to measure the global net performance.

Theoretical and numerical results proved that quasi-Newton algorithms are superior to steepest gradient algorithms (Dennis and Schnabel, 1983). For this reason, several researchers proposed these techniques to train neural nets. Watrous (1987) employed DFP and BFGS methods and compared them with the backpropagation algorithm; this comparison showed that DFP and BFGS methods need fewer iterations, but each iteration required the update of the Hessian approximation and more calculation time. Parker (1987) derived a formula for the update of the inverse Hessian that suits a parallel implementation. Becker and Le Cun (1988) proposed simple diagonal approximations of the Hessian which, however, led to no significant time reduction for convergence. Kollias and Anastassiou (1989) modified the Marquardt–Levenberg method to use only a near-diagonal form of the approximated Hessian matrix, which allowed a parallel computation for the solution of the linear system. In a serial computation, the method does not provide a clear gain in calculation time. Because of these limitations, Barnard (1992) proposed a stochastic

† To whom all correspondence should be addressed.

method and noticed that his method was better than deterministic methods such as conjugate gradient and variable metrics. Bello (1992) used a nonlinear least-squares optimization algorithm enhanced with a quasi-Newton algorithm for performing additional iterations of the "global-batch" optimization problem.

Many authors stated that quasi-Newton methods are limited to middle-sized applications because of the computation time and the memory space needed to perform the update of the Hessian approximation (Watrous, 1987; Nahas *et al.*, 1992). This paper proposes a modification to the classical approach of the quasi-Newton method. The modification consists of a new Hessian approximation that neglects some second-order interactions and which is built by taking into account the structure of the network. More specifically, the number of elements in the approximated Hessian is no longer the square of the total number of weights in the network but is, instead, proportional to the number of neurons in each level. Since our approach limits the number of elements included in the approximated Hessian dimension, it performs the training phase of a neural net without the computation time and the memory space problem which is usually associated with quasi-Newton methods. The paper begins with the presentation of the mathematical formulation used for three classical approaches: the backpropagation, the conjugate gradient and the quasi-Newton method. The next section outlines the Hessian approximation and the three proposed modifications to this approximation. The new methods are then compared with the classical approaches on two testing problems. In conclusion, an analysis and discussion of results will follow.

2. MATHEMATICAL FORMULATION

2.1. Notation

The following models are developed for fully connected feedforward neural networks with a single hidden layer. The purpose of using a single hidden layer is for notation simplicity as the extension of each model to several layers is straightforward.

Figure 1 shows the structure of the network. The input, hidden and output layers are noted $[X, H, O]$ respectively while the output training patterns are noted Y; $[x, h, o]$ are the total number of neurons in each layer $[X, H, O]$; $W^{(L)}$ are the weights in level L and $f^{(L)}$ is the activation function for that level. With one hidden layer, $L = 1$ or 2. X^0 and H^0, the bias for the corresponding layer, are always equal to one.

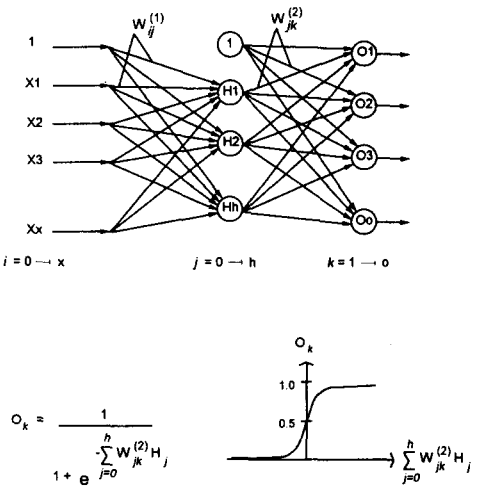

Fig. 1. Schematic of a feedforward neural network.

The neurons H_{pj} and O_{pk} are calculated via activation functions of the forms:

$$H_{pj} = f^{(1)}\left(\sum_{i=0}^{x} W_{ij}^{(1)} X_{pi}\right) \tag{1}$$

$$O_{pk} = f^{(2)}\left(\sum_{j=0}^{h} W_{jk}^{(2)} H_{pj}\right). \tag{2}$$

The training is done by presenting P presentations of input–output vectors $[X_p, Y_p]$ to the neural network and by minimizing a function of Y and O of the form:

$$E = \sum_{p=1}^{P} \sum_{k=1}^{o} Ep(Y_{pk}, O_{pk}), \tag{3}$$

where Ep is an error function.

2.1.1. Gradient evaluation. Let i, j and k be the elements of the corresponding layer $[X, H, O]$ and i', j' be the elements of any two successive layers. Define an error function as:

$$Ep = \frac{1}{2} \sum_{k} (Y_{pk} - O_{pk})^2 \tag{4}$$

for one presentation p, so that:

$$E = \sum_{p} Ep. \tag{5}$$

Define

$$\text{sum}_{pj} = \sum_{i} W_{ij}^{(1)} X_{pi} \tag{6}$$

and

$$\text{sum}_{pk} = \sum_{j} W_{jk}^{(2)} H_{pj}. \tag{7}$$

The gradient is then expressed for the final layer by:

$$\frac{\partial E}{\partial W_{jk}^{(2)}} = -\sum_p f'^{(2)}(\text{sum}_{pk})(Y_{pk} - O_{pk})H_{pj} \quad (8)$$

and for the hidden layer by:

$$\frac{\partial E}{\partial W_{ij}^{(1)}} = -\sum_p f'^{(1)}(\text{sum}_{pj}) \times \sum_k [f'^{(2)}(\text{sum}_{pk})(Y_{pk} - O_{pk})W_{jk}^{(2)}]X_{pi}. \quad (9)$$

Rewriting the gradients in a compact notation with an iteration index (n) will give for the gradient associated to one weight and for one presentation:

$$Gp_{i'j'}^{(L)}(n) = \frac{\partial Ep}{\partial W_{i'j'}^{L}(n)} \quad (10)$$

and for the sum of gradients for one weight:

$$G_{i'j'}^{(L)}(n) = \sum_p \frac{\partial Ep}{\partial W_{i'j'}^{L}(n)}. \quad (11)$$

We define $\mathbf{G}(n)$ as the gradient vector whose elements $G_m(n)$ are all the $G_{i'j'}^{(L)}(n)$ (**bold** characters refer to vectors).

$$G_m(n) = G_{i'j'}^{(L)}(n) \quad (12)$$

$$m = (j'-1)(x+1) + i' + 1$$
$$1 \leqslant m \leqslant (x+1)h$$
$$i' \in [0, x]$$
$$j' \in [1, h]$$

$$m = (j'-1)(h+1) + i' + 1 + (x+1h$$
$$(x+1)h < m \leqslant (x+1)h + (h+1)o$$
$$i' \in [0, h]$$
$$j' \in [1, o].$$

2.2. *The classical methods*

2.2.1. Backpropagation. The simplest minimizing technique, when the gradient is available, is to select the steepest descent direction and to apply a unidirectional search along this direction. This steepest descent optimization technique is defined by

$$\Delta\mathbf{W}(n) = -\lambda(n)\mathbf{G}(n) \quad (13)$$

where $\Delta\mathbf{W}(n)$ is the variation of the weights for one iteration and $\lambda(n)$ is a coefficient minimizing the function in the descent direction. It has been shown that there is no advantage in finding the exact minimum on the search direction at each iteration (Dennis and Schnabel, 1983). The backpropagation algorithm proposed by Werbos (1974) and popularized by Rumelhart *et al.* (1986) is based on a variation of the steepest descent technique. There is no unidirectional search used but a fixed descent step, η, called learning rate which is added to a fraction of the last variation, α, called momentum. The last term introduces some elements of conjugate gradient method. This algorithm is defined by:

$$\Delta_p W_{i'j'}^{(L)}(n) = -\eta Gp_{i'j'}^{(L)}(n) + \alpha\Delta_p W_{i'j'}^{(L)}(n-1) \quad (14)$$

when doing a continuous update and by:

$$\Delta\mathbf{W}(n) = -\eta\mathbf{G}(n) + \alpha\Delta\mathbf{W}(n-1) \quad (15)$$

when a batch update procedure is chosen.

2.2.2. Conjugate gradient method. This method, proposed by Fletcher and Reeves (1964), uses successive conjugate directions based on the gradient and the residue. If the objective function is quadratic and the search direction is minimized exactly at each iteration, the method has quadratic convergence property. It offers a major improvement over steepest descent methods with only a small increase in the required computational effort. The method has been used by Leonard and Kramer (1990) to train neural networks as an alternative to backpropagation. The weights of a network are updated according to a unidirectional search in the descent direction $\mathbf{S}(n)$ by:

$$\Delta\mathbf{W}(n) = \lambda(n)\mathbf{S}(n) \quad (16)$$

and the descent direction $\mathbf{S}(n)$ is computed from the gradient of past and present iterations, by

$$\mathbf{S}(n) = \mathbf{G}(n) + \frac{\|\mathbf{G}(n)\|}{\|\mathbf{G}(n-1)\|}\mathbf{S}(n-1). \quad (17)$$

For the descent directions to remain truly conjugate, the unidirectional search, at every iteration, has to be carried at high precision. Nevertheless, after many iterations, the directions might become nearly parallel; to overcome this difficulty, Fletcher and Reeves (1964) suggested to restart the procedure by equalling the descent direction to the gradient at every $(t+1)$ iteration, where (t) is the total number of weights in the network.

2.2.3. Quasi-Newton method. By calculating the second derivatives of the objective function, we obtain a better understanding of the function topology, which leads in turn to choose a more efficient descent direction. Let:

$$\Delta\mathbf{W}(n) = \lambda(n)\mathbf{S}(n) \quad (18)$$

where the descent direction $\mathbf{S}(n)$ is defined by:

$$\mathbf{S}(n) = -[\mathbf{H}(n)]^{-1}\mathbf{G}(n) \quad (19)$$

and where $\mathbf{H}(n)$ is the Hessian matrix. The main difficulty with this approach is that to find the solution of this system at every iteration is a very tedious task. The variable metric methods, also

called quasi-Newton methods, bypass this difficulty by directly approximating the inverse Hessian matrix, $[\tilde{\mathbf{H}}(n)]^{-1}$, from the first derivative, $\mathbf{G}(n)$. These methods are the most popular unconstrained optimization techniques and, among them, BFGS is the most widely used method. It calculates $[\tilde{\mathbf{H}}(n)]^{-1}$ by:

$$\Delta[\tilde{\mathbf{H}}(n)]^{-1} = [\tilde{\mathbf{H}}(n+1)]^{-1} - [\tilde{\mathbf{H}}(n)]^{-1} \quad (20)$$

$$\Delta[\tilde{\mathbf{H}}(n)]^{-1} = \frac{[\Delta\mathbf{W}(n) - [\tilde{\mathbf{H}}(n)]^{-1}\Delta\mathbf{G}(n)][\Delta\mathbf{W}(n)]^T + [\Delta\mathbf{W}(n)][\Delta\mathbf{W}(n) - [\tilde{\mathbf{H}}(n)]^{-1}\Delta\mathbf{G}(n)]^T}{[\Delta\mathbf{G}(n)]^T\Delta\mathbf{W}(n)} - \frac{[\Delta\mathbf{W}(n) - [\tilde{\mathbf{H}}(n)]^{-1}\Delta\mathbf{G}(n)]^T\Delta\mathbf{G}(n) \times \Delta\mathbf{W}(n)[\Delta\mathbf{W}(n)]^T}{[[\Delta\mathbf{G}(n)]^T\Delta\mathbf{W}(n)][[\Delta\mathbf{G}(n)]^T\Delta\mathbf{W}(n)]}. \quad (21)$$

3. A SIMPLIFIED FORM OF THE HESSIAN MATRIX

We developed three methods based on a simplification of the terms included in the approximated Hessian matrix, each one neglecting a different number of interactions depending on the structure of the network. Neglecting second-order interactions means a less efficient descent direction in the quasi-Newton method, which will increase the number of iterations required to reach a solution. This effect will be compensated by a reduction in the computation time needed to perform the update of the approximated Hessian matrix and by a decrease in the required memory space. By looking at three different combinations, we are seeking to find the best compromise between the size of the approximated Hessian matrix and the computation time necessary to optimize the weights in a network. Our first method neglects the second-order interactions between weights of different levels and considers a separate matrix **H** for each level (BFGS-L). Bishop (1992) has shown that the real Hessian matrix of a multilayer neural network is sparse with respect to the final layer, which means that there is no interaction among the output neurons. The second method (BFGS-M) takes this knowledge into account and uses a separate matrix **H** for each output neuron and a matrix **H** for the hidden layer. The third method assumes that only the weights connected to the same neuron have important second-order interaction and it associates a matrix **H** to every output and hidden neuron (BFGS-N). The "M" in BFGS-M stands for mixed, as this method is a combination both BFGS-L and BFGS-N methods. The BFGS-M method is equivalent to the BFGS-L method when there is only one output neuron.

The major advantage in associating the Hessian matrix to a level or neuron is that it reduces considerably the total size of the matrix to be calculated.

Figure 2 shows the size of the matrix applied to a typical neural network for the following configurations: global, by level, mixed and by neuron. The neural network has two inputs, two neurons in the hidden layer and two outputs. For the illustrated network, the matrix **H** with the global configuration (a) holds $(12)^2 = 144$ elements while the level configuration (b) holds $(6)^2 + (6)^2 = 72$; the mixed configuration (c) holds $(6)^2 + (3)^2 + (3)^2 = 54$ elements; and the neuron configuration (d) holds $(3)^2 + (3)^2 + (3)^2 + (3)^2 = 36$ elements. For example, the second derivative $\partial^2 E/\partial W_5 \partial W_9$ does not need to be calculated in configurations (b), (c) and (d). The actual approximated Hessian matrix is symmetric and only the upper part needs to be computed, but the previous calculations remain relevant for the size reduction.

3.1. Search vector expression

In this approach, the formulation of the weights variation is similar to quasi-Newton methods. It differs in the elements selected to form the gradient vector. In particular, for the weights variation by level, let $\mathbf{G}^{(L)}(n)$ be the vectors whose elements are all the $G_{ij}^{(L)}(n)$ of a single level, where:

$$\mathbf{G} = [\mathbf{G}^{(1)}, \mathbf{G}^{(2)}] \quad (22)$$

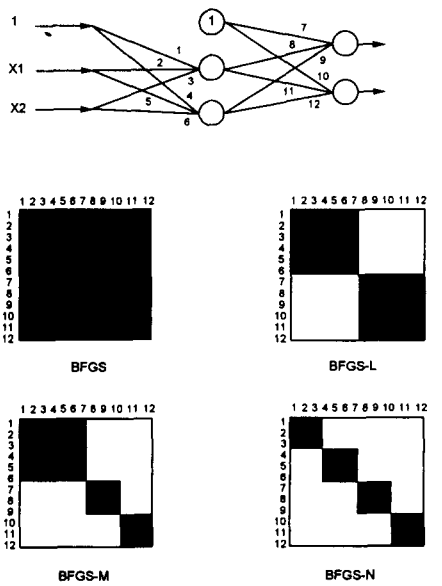

Fig. 2. Schematic of second order interaction. The gray zones represent the dimensions of the Hessian matrix for different configurations: (a) global (BFGS); (b) by level (BFGS-L); (c) by neuron (BFGS-N); and (d) the mixed method (BFGS-M).

and where $\mathbf{G}^{(1)}$ holds $(x+1)h$ elements and $\mathbf{G}^{(2)}$ holds $(h+1)o$ elements. The weights variation for each level will then be defined by:

$$\Delta \mathbf{W}^{(L)}(n) = \lambda(n)\mathbf{S}^{(L)}(n) \tag{23}$$

with a descent direction for each level defined by:

$$\mathbf{S}^{(L)}(n) = -[\mathbf{H}^{(L)}(n)]^{-1}\mathbf{G}^{(L)}(n). \tag{24}$$

The two Hessian matrices, $\mathbf{H}^{(L)}$, are symmetric matrix of dimension $(x+1)h$ and $(h+1)o$, respectively.

In the same way, the weights variation may be rewritten as a function of neurons. Let $\mathbf{G}_j^{(L)}(n)$ be the vector whose elements are all the $\mathbf{G}_{ij}^{(L)}(n)$ connected to the same neuron and where:

$$\mathbf{G}^{(1)} = [\mathbf{G}_1^{(1)}, \mathbf{G}_2^{(1)}, \ldots, \mathbf{G}_j^{(1)}, \ldots, \mathbf{G}_h^{(1)}] \tag{25}$$

$$\mathbf{G}^{(2)} = [\mathbf{G}_1^{(2)}, \mathbf{G}_2^{(2)}, \ldots, \mathbf{G}_j^{(2)}, \ldots, \mathbf{G}_o^{(2)}] \tag{26}$$

$\mathbf{G}_j^{(1)}$ holds $(x+1)$ elements and $\mathbf{G}_j^{(2)}$ holds $(h+1)$ elements. The weights variation is then defined by:

$$\Delta \mathbf{W}_j^{(L)}(n) = \lambda(n)\mathbf{S}_j^{(L)}(n) \tag{27}$$

$$\mathbf{S}_j^{(L)}(n) = -[\mathbf{H}_j^{L)}(n)]^{-1}\mathbf{G}_j^{(L)}(n). \tag{28}$$

In these expressions, the matrices $\mathbf{H}_{j(1)}$ are symmetric with dimension $(x+1)$ and the matrices $\mathbf{H}_{j(2)}$ are symmetric with dimension $(h+1)$. Each of these matrices can be computed independently during each iteration in a distributed parallel form.

4. RESULTS AND DISCUSSION

To evaluate the performance of the proposed modifications, six algorithms are compared through two examples. The six algorithms are backpropagation (BP), conjugate gradient (CG), quasi-Newton with BFGS update and modified quasi-Newton methods with respectively BFGS-L (the update proposed as a function of levels in the network), BFGS-M (the method which is both a function of neurons and hidden levels) and BFGS-N (the update proposed as a function of neurons). For BP, a batch update procedure was chosen. For all the BFGS algorithms, $\lambda(n)$, the descent step, is evaluated by a unidirectional search method, following the Armijo rule (Armijo, 1966). We have to note that all the BFGS algorithms coded here do not apply any corrections to the Hessian matrix when these are positive semi-definite, a process that is very time-consuming. Instead, when the updated Hessian matrix is found to be positive semidefinite, this matrix is replaced by the identity matrix. For the conjugate gradient algorithm, the descent step, $\lambda(n)$, is evaluated to a relative precision of 10^{-9}. A sigmoid function is used as an activation function throughout the algorithms. Ten sets of initial weights were generated randomly and the six algorithms were each tried with all the sets. Calculations were done using an IBM RS/6000 Unix-based computer. All the algorithms were coded using the "Basic Linear Algebra Subroutines", or BLAS, which provides a high level of performance for large linear algebraic equations. The internal coding of the sub-routines are platform-specific so that the code is tuned to get the best performance from the computer architecture's characteristics.

4.1. *Bioreactor example*

The algorithms were first tested on a model giving way to complex dynamics. It derives from a continuous flow stirred tank reactor (CFSTR) in which cell growth is inhibited by large substrate concentration and for which Agrawal *et al.* (1982) obtained the basic equations. This system is suggested as a benchmark for adaptive network-based process control by Ungar (1990). The dimensionless mass balance equations of the system are:

$$\frac{dC_1}{dt} = -C_1 + DaC_1(1-C_2)\,e^{C_2/\gamma} \tag{29}$$

$$\frac{dC_2}{dt} = -C_2 + DaC_1\frac{1+\beta}{1+\beta-C_2}(1-C_2)\,e^{C_2/\gamma} \tag{30}$$

where C_1 and C_2 are dimensionless cell mass and substrate conversion, B is a parameter related to the yield coefficient, γ is a parameter related to the maximum specific growth rate, Da is the Damkohler number which is written as the specific growth rate at the inlet concentration over the dilution rate τ is the time multiplied by the dilution rate. When $\beta = 0.02$ and $\gamma = 0.48$, a Hopf bifurcation point occurs at $Da = 1.206$, beyond which the long-term behavior of the system develops into a limit cycle.

The neural net is used to model the dynamic of the system in the operating region of $Da = [1.2\text{–}2.0]$. The training database was developed by generating a random signal of Da in the corresponding region (Fig. 3) and by integrating the model at an interval of $\tau = 0.1$, with starting point at $Da = 1.6$ and the corresponding steady state for C_1 and C_2. The network layout is made of an input layer with nine neurons representing Da, C_1 and C_2 at time (t), $(t-1)$ and $(t-2)$, a hidden layer with 30 neurons and an output layer of two neurons for the prediction of C_1 and C_2 at $(t-1)$. The training set contains 40 presentations. A weights set is considered at a solution when the total error E is less than 10^{-2}. For the backpropagation algorithm, the best results were obtained with $\eta = 0.7$ and $\alpha = 0.9$.

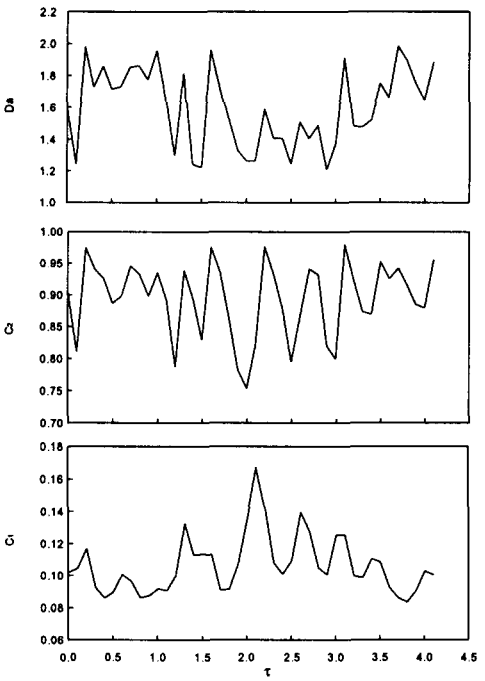

Fig. 3. Graph of the training database for the bioreactor model.

Table 1 presents the results for the bioreactor test problem. For each method, it shows the number of cases that converge, the average iteration count, the average computation in seconds and relative to the BFGS-N method, and, for the quasi-Newton methods, the dimension of the approximated Hessian matrix. Cases where the method failed to converge in less than 2000 iterations (20,000 for BP) were excluded from the statistics. The average iteration count shows that the more elements that are included in the approximated Hessian matrix, the better the method performs. In term of iterations, the conjugate gradient method is between the BFGS-M and the BFGS-N methods. Since the time required per iteration varies greatly among the different methods, the most significant result is the average time necessary to reach a solution.

The best result was obtained with the proposed BFGS-N method followed closely by the standard

Table 1. Results of the bioreactor test problem

Bioreactor method	Conv./ 10	Iterations Avg	Time Avg	Time Relative	Hessian size
BP	9	10,174	189.3	6.44	
CG	10	731	105.4	3.59	
BFGS	10	232	32.5	1.11	131,044
BFGS-L	10	425	45.7	1.55	93,844
BFGS-M	10	553	59.4	2.02	91,922
BFGS-N	9	863	29.4	1	4922

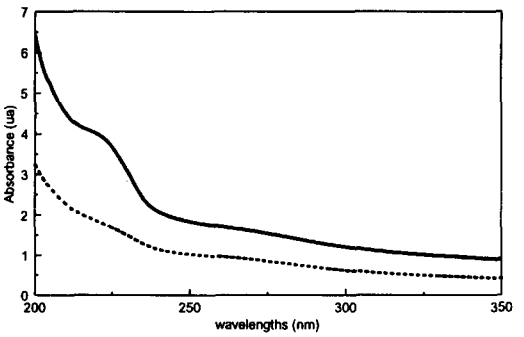

Fig. 4. Example of two spectra used for the identification of suspended solids (SS) and chemical oxygen demand (COD) of wastewater (——: SS = 182 mg/l; COD = 633 mg/l O_2; – – –: SS = 68 mg/l; COD = 199 mg/l O_2).

BFGS method. All four quasi-Newton methods outperform the conjugate gradient and the backpropagation methods. In the present network configuration, the reduction in memory required for the BFGS-N method compared to the standard BFGS method represents a clear improvement, while it is only marginal for the BFGS-L and BFGS-M methods.

4.2. Ultraviolet multi-wavelength absorptiometry analysis

Spectra analysis is an interesting and challenging domain for neural networks. Their large number of inputs and their highly nonlinear characteristics make them a very appropriate test for the training algorithms. In the second test case, a neural network is used to correlate ultraviolet (UV) absorption spectra of wastewater and global water quality parameters such as suspended solids (SS) and chemical oxygen demand (COD). Thomas and Galot (1990) have shown that many water quality parameters are related to the UV absorption spectrum in the wavelength range of 200–350 nm. These spectra are highly influenced by the presence of different components such as nitrate, chromium and phenol. With mathematical treatment applied to a spectrum, they were able to predict, with good precision, the concentration of these components in wastewater. More recently, Renard (1995) has been studying the use of neural nets to produce a model that can estimate, from the UV spectrun, more global water quality parameters such as biological oxygen demand (BOD), COD, total organic carbon (TOC), SS, etc. These networks are very large and sometimes require that the spectrum used at input be discretized into 150 wavelengths.

Figure 4 presents two typical spectra used in this study. The network used in the tests has 51 inputs, each representing the UV absorption at a different

Table 2. Results of the ultraviolet absorption spectra test problem

UV spectrum	Conv./ 10	Iterations Avg	Time Avg	Time Relative	Hessian size
BP	9	28,671	525.2	13.50	
CG	10	1769	96.8	2.49	
BFGS	9	249	237.2	6.10	1,170,724
BFGS-L	9	406	416.0	10.69	1,083,364
BFGS-M	8	450	450.1	11.78	1,082,482
BFGS-N	10	479	38.9	1	54,962

wavelength of the spectrum in the 200–350 nm range, 20 hidden neurons and two output neurons expressing the SS and COD values of the water sample. The training set is made of 15 spectra, covering the SS range of 8–436 mg/l and the COD range of 25–888 mg/l O_2. Calculation is stopped when the total output error E is less than 10^{-3}. The learning rate, η, and the momentum, α, of the backpropagation algorithm were fixed to 0.3 and 0.7 respectively.

The results for this test problem are found in Table 2. Cases where the method failed to converge in less than 3000 iterations (50,000 for BP) were excluded from the statistics. The iteration average needed by the three proposed modifications is roughly the same and is close to twice the amount necessary for the BFGS method, while the conjugate gradient method and the backpropagation method require significantly more. Once again, in terms of computation time, BFGS-N is the fastest method. The average time per iteration for the conjugate gradient method is less than for the BFGS-N method, but the overall peformance of the BFGS-N is 2.5 times better than the CG method. For the three methods storing large Hessian matrix, BDGS, BFGS-L and BFGS-M, the average time per iteration is relatively large as heavy memory management impedes computing efficiency even when the algorithms are coded using high-performance linear algebra sub-routines. It should be noted that, in this example, backpropagation is not much slower than the BFGS-L or the BFGS-M method. In this example, the space requirement of the BFGS-N method is ~one-twentieth of the required for the other quasi-Newton methods tested.

CONCLUSON

Three modifications to the classical approach of the quasi-Newton method have been presented. It was shown that the hypotheses supporting those methods are relevant and desirable in terms of convergence rate. The BFGS-N method, the proposed update as a function of neurons, is superior when compared to regular quasi-Newton method such as BFGS, to the conjugate gradient method and to the backpropagation algorithm. It represents a clear gain in terms of computational time. This is done without a major increase in memory space required when compared to the backpropagation or the conjugate gradient algorithms, making the approach suitable for large-scale problems. There is also no need to adjust parameters (compared to backpropagation) which makes this algorithm easy to use. This algorithm is also easy to implement in a distributed parallel form. The BFGS-L and the BFGS-M algorithms, in the network configurations tested, did not present good compromises in terms of memory space and computational time.

Acknowledgement—This research project was partially funded by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

Agrawal P., C. Lee, H. C. Lim and D. Ramkrishna, Theoretical investigations of dynamic behavior of isothermal continuous stirred tank biological reactors. *Chem. Engng Sci.* **37**, 453–562 (1982).

Armijo L., Minimization of functions having Lipschitz-continuous first partial derivatives. *Pac. J. Math.* **16**, 1–3 (1966).

Barnard E., Optimization for training neural nets. *IEEE Trans. Neural Networks* **3**, 232–240 (1992).

Becker S. and Y. Le Cun, Improving the convergence of back-propagation learning with second order methods. In D. Touretzky, G. Hinton and T. Sejnowski (eds). *Proc. Connectionist Model Summer School*, pp. 29–37. Morgan-Kaufman, San Mateo (1988).

Bello M. G., Enhanced training algorithms and integrated training/architecture selection for multilayer perceptron networks. *IEEE Trans. Neural Networks* **3**, 864–875 (1992).

Bishop C., Exact calculation of the Hessian matrix for the multilayer perceptron. *Neural Comput.* **4**, 494–501 (1992).

Dennis J. E. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimisation and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, New Jersey (1983).

Fahlman S. E., An empirical study of learning speed in back-propagation networks. Internal report: CMU-CS-88/162, Carnegie Mellon University, Pittsburgh (1988).

Fletcher R. and C. M. Reeves, Function minimization by conjugate gradients. *Comput. J.* **7**, 149–154 (1964).

Hoskins J. C. and D. M. Himmelblau, Artificial neural network model of knowledge representation in chemical engineering. *Computers chem. Engng* **12**, 881–890 (1988).

Jacobs R. A., Increased rates of convergences through learning rate adaptation. *Neural Networks* **1**, 29–36 (1988).

Kollias S. and D. Anastassiou, An adaptive least squares algorithm for the efficient training of artificial neural networks. *IEEE Trans. Circuits Systems* **36**, 1092–1101 (1989).

Leonard J. A. and M. A. Kramer, Improvement of the backpropagation algorithm for training neural networks. *Computers chem. Engng* **14**, 337–341 (1990).

Nahas E. P., M. A. Henson and D. E. Seborg, Nonlinear internal model control strategy for neural network models. *Computers chem. Engng* **16**, 1039–1057 (1992).

Parker D. B., Optimal algorithms for adaptive networks: second order back propagation, second order direct

propagation, and second order Hebbian learning. *IEEE First Int. Conf. Neural Networks*, San Diego, II-593-600 (1987).

Renard F., Réseaux de neurones couplés à la spectrophotométrie ultraviolette pour le contrôle automatique de la qualité des eaux. Master thesis, Université de Sherbrooke, Sherbrooke, Québec, Canada, to be published Summer 1995.

Rigler A. K., J. M. Irvine and K. Vogl, Rescaling of variables in backpropagation learning. *Neural Networks* **4**, 225–229 (1991).

Rumelhart D. E. G. E. Hinton and R. J. Williams, Learning internal representation by error propagation. In D. E. Rumelhart and J. L. McClelland (eds), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Ch. 8. MIT Press, Cambridge, MA (1986).

Tallaneare T., SuperSAB: fast adaptive backpropagation with good scaling properties. *Neural Networks* **3**, 561–573 (1990).

Thomas O. and S. Gallot, Ultaviolet multiwavelength absorptiometry (UVMA) for the examination of natural waters and wastewaters; part I: general considerations. *Fresenius J. Anal. Chem.* **338**, 234–237 (1990).

Van Ooyen A. and B. Nienhuis, Improving the convergence of the back-propagation algorithm. *Neural Networks* **5**, 465–471 (1992).

Ungar L. H., A bioreactor benchmark for adaptive network-based process control. In W. T. Miller, III, R. S. Sutton and P. J. Werbos (eds), *Neural Networks for Control*, Ch. 16. MIT Press, Cambridge, MA, U.S.A. (1990).

Watrous R. L., Learning algorithms for connectionist networks: applied gradient methods of nonlinear optimization. *IEEE First Int. Conf. Neural Networks*, San Diego, II-619-627 (1987).

Werbos P. J., Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. Thesis, Harvard University Committee in Applied Mathematics (1974).