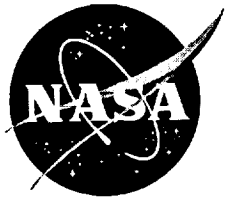
NASA/TM-1998-112239



Accelerated Training for Large Feedforward Neural Networks

Slawomir W. Stepniewski and Charles C. Jorgensen

November 1998

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at *http://www.sti.nasa.gov*

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at (301) 621-0134

- Telephone the NASA Access Help Desk at (301) 621-0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-1998-112239

Accelerated Training for Large Feedforward Neural Networks

Slawomir W. Stepniewski
Ames Research Center, Moffett Field, California

Charles C. Jorgensen
Ames Research Center, Moffett Field, California

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035-1000

November 1998

Available from:

NASA Center for AeroSpace Information
7121 Standard Drive
Hanover, MD 21076-1320

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Accelerated Training for Large Feedforward Neural Networks

SLAWOMIR W. STEPNIEWSKI AND CHARLES C. JORGENSEN

Summary

In this paper we introduce a new training algorithm, the scaled variable metric method. Our approach attempts to increase the convergence rate of the modified variable metric method. It is also combined with the RBackprop algorithm, which computes the product of the matrix of second derivatives (Hessian) with an arbitrary vector. The RBackprop method allows us to avoid computationally expensive, direct line searches. In addition, it can be utilized in the new, "predictive" updating technique of the inverse Hessian approximation. We have used directional slope testing to adjust the step size and found that this strategy works exceptionally well in conjunction with the RBackprop algorithm. Some supplementary, but nevertheless important, enhancements to the basic training scheme such as, improved adjustment of a scaling factor for the variable metric update and computationally more efficient procedure for updating the inverse Hessian approximation are presented as well. We summarize by comparing the scaled variable metric method with four first- and second-order optimization algorithms, including a very effective implementation of the Levenberg-Marquardt method. Our tests indicate promising computational speed gains of the new training technique, particularly for large feedforward networks, i.e., for problems where the training process may be the most laborious.

1. Introduction

For some neural network applications requiring high modeling/mapping accuracy, it may not be sufficient to employ first order training methods based on the gradient descent schemes with adjustable learning rates. Although these training algorithms are relatively inexpensive computationally, they could perform poorly, because the search directions can partially overlap and interfere with each other producing the undesirable effect of impairing previous minimization efforts during subsequent iterations (refs. 1 & 2). Moreover, for problems with rapid changes of the objective function, small variations in the step sizes may result in considerably different gradient directions and even entire search paths.

In optimization theory, several solutions have been proposed to boost the effectiveness of consecutive directional searches (ref. 1). Conjugate gradient methods attempt to construct non-interfering directions based on a steady quadratic model of the objective function and the assumption of exact line searches along those search directions. The more effective Newton and trust-region methods rely on a more detailed quadratic model of the merit function rederived at each iteration. A serious drawback of these algorithms is the significant computational overhead of obtaining the matrix of second partial derivatives (Hessian) or its approximation.

Quasi-Newton (secant) training methods also utilize a Hessian approximation or its inverse. The computational efficiency of quasi-Newton methods comes from the fact that a Hessian approximation is continuously built during function minimization. The updating process is substantially faster than computing a complete Hessian matrix. However, the lack of precise knowledge of second derivatives may have a negative impact on the training convergence rate. In practice, the algorithm may also be more susceptible to local minima and round-off errors. It may be advantageous, nevertheless, to use quasi-Newton methods for problems where other second order algorithms are computationally too expensive and gradient descent methods produce unsatisfactory results. In this paper, we present a new variation of one quasi-Newton method, the scaled variable metric (SVM) method. The method appears to be quite competitive with other leading training techniques. Our tests show that for large neural networks having more than several hundred weights, the SVM technique typically outperforms standard variable metric algorithms in convergence speed and in some cases it is also able to produce the most accurate neural models.

2. Quasi-Newton Methods

Although quasi-Newton optimization algorithms are most commonly understood as techniques to construct successive Hessian approximations or their inverses, many modern approaches view them primary as strategies to choose a series of search directions. These methods put less emphasis on the issue of convergence to the true Hessian. The majority of quasi-Newton

optimization algorithms utilize formulae that build inverse Hessian approximations (denoted as matrix **D**). Methods constructing direct Hessian approximations (refs. 3 & 4) are used less frequently. Our algorithm belongs to the group of variable metric methods, an important subclass of quasi-Newton algorithms that ensure positive definiteness of **D**. The Huang formula (refs. 4 & 5) defined by

$$\mathbf{D}_{k+1} = \mathbf{D}_k + \Delta\mathbf{D}_k = \mathbf{D}_k + \rho\frac{\Delta\mathbf{w}_k\left(K_1\Delta\mathbf{w}_k + K_2\mathbf{D}_k^{\mathrm{T}}\Delta\mathbf{g}_k\right)^{\mathrm{T}}}{\left(K_1\Delta\mathbf{w}_k + K_2\mathbf{D}_k^{\mathrm{T}}\Delta\mathbf{g}_k\right)^{\mathrm{T}}\Delta\mathbf{g}_k} - \frac{\mathbf{D}_k\Delta\mathbf{g}_k\left(L_1\Delta\mathbf{w}_k + L_2\mathbf{D}_k^{\mathrm{T}}\Delta\mathbf{g}_k\right)^{\mathrm{T}}}{\left(L_1\Delta\mathbf{w}_k + L_2\mathbf{D}_k^{\mathrm{T}}\Delta\mathbf{g}_k\right)^{\mathrm{T}}\Delta\mathbf{g}_k} \tag{1}$$

is a fairly general update for which the well-known BFGS (Broyden-Fletcher-Goldfarb-Shanno) and DFP (Davidon-Fletcher-Powell) formulae are special cases. In (1) $\Delta\mathbf{w}_k = \mathbf{w}_{k+1} - \mathbf{w}_k$ is the vector of weight corrections and $\Delta\mathbf{g}_k$ denotes the corresponding gradient change. The ρ parameter must be positive to preserve positive definite property of $\mathbf{D}_k$. The four other scalars K_1, K_2, L_1, and L_2, can be chosen arbitrarily except for $L_1 = 0$ and $L_2 = 0$ at the same time. Note that (1) allows $\mathbf{D}_k$ to be unsymmetric. When updating $\mathbf{D}_k$ using (1), at every iteration the following condition is satisfied

$$\mathbf{D}_{k+1}\Delta\mathbf{g}_k = \rho\Delta\mathbf{w}_k \tag{2}$$

An interesting property of the Huang update is that for strictly quadratic error function $E(\mathbf{w})$ with a positive definite Hessian $\mathbf{H} = \partial^2 E/\partial\mathbf{w}^2$ and the initial matrix $\mathbf{D}_0$ chosen so $(\mathbf{D}_0 + \mathbf{D}_0^{\mathrm{T}})/2$ is also positive definite, the series $\mathbf{D}_k \to \rho\mathbf{H}^{-1}$ when ρ is constant (refs. 5 & 4). In practice, most variable metric algorithms which utilize (1) with $\rho \neq 1$ do not preserve a constant ρ but rather attempt to tune it on-line. Nevertheless, convergence to $\rho\mathbf{H}^{-1}$ is an appealing property and limiting frequent and large changes of this scaling factor may be beneficial.

The Huang family of formulae offers an infinite number of choices for its adjustable parameters with minimal theoretical background as to how best set them optimally. In fact, for the strictly quadratic error function and exact line searches, the sequence of search directions is independent of the particular choice of K_1, K_2, L_1, L_2, and constant ρ (ref. 4). However, for non-quadratic functions and inexact line searches, different updating formulas derived from (1) are not equivalent. In this paper we consider the following choice for the K_i and L_i (i =1, 2) parameters

$$K_1 = 1 - \frac{1}{\rho}\frac{\Delta\mathbf{g}_k^{\mathrm{T}}\mathbf{D}_k\Delta\mathbf{g}_k}{\Delta\mathbf{w}^{\mathrm{T}}\Delta\mathbf{g}_k} \qquad K_2 = -\frac{1}{\rho}$$
$$L_1 = \frac{\Delta\mathbf{g}_k^{\mathrm{T}}\mathbf{D}_k\Delta\mathbf{g}_k}{\Delta\mathbf{w}_k^{\mathrm{T}}\Delta\mathbf{g}_k} \qquad L_2 = 0 \tag{3}$$

Substituting (3) in (1) leads to the expression

$$\mathbf{D}_{k+1} = \mathbf{D}_k + \Delta\mathbf{D}_k = \mathbf{D}_k + \frac{\rho}{b_k}\Delta\mathbf{w}_k\Delta\mathbf{w}_k^{\mathrm{T}} - \frac{1}{a_k}(\mathbf{D}_k\Delta\mathbf{g}_k)(\mathbf{D}_k\Delta\mathbf{g}_k)^{\mathrm{T}} + a_k\mathbf{r}_k\mathbf{r}_k^{\mathrm{T}} \tag{4}$$

which produces symmetric updates. In equation (4) $a_k = \Delta\mathbf{g}_k^{\mathrm{T}}\mathbf{D}_k\Delta\mathbf{g}_k$, $b_k = \Delta\mathbf{w}_k^{\mathrm{T}}\Delta\mathbf{g}_k$ and $\mathbf{r}_k = \Delta\mathbf{w}_k/b_k - \mathbf{D}_k\Delta\mathbf{g}_k/a_k$ are introduced for notational convenience. Because (4) is very similar to the BFGS formula except for the ρ scalar, this equation will be referred as the extended or modified BFGS formula. Various authors have tried to find the optimal setting for the ρ parameter (see ref. 3 for a review). It is the paradigm proposed and studied by Oren (ref. 6) which, in our view, offers an elegant mathematical justification of how to assess the useful range of ρ values.

3. Convergence Acceleration

The idea that the convergence rate of the variable metric methods could be controlled was originally suggested by Oren and Luenberger in the context of the self-scaling variable metric method (ref. 6). The concept is based on the theorem, which considers the positive definite quadratic form

$$F(\mathbf{x}) = \frac{1}{2}\left(\mathbf{x} - \mathbf{x}^*\right)^{\mathrm{T}}\mathbf{H}\left(\mathbf{x} - \mathbf{x}^*\right) \tag{5}$$

with respect to $(\mathbf{x} - \mathbf{x}^*)$ and an abstract optimization algorithm which aims to find the stationary point $\mathbf{x}^*$. It is assumed that the minimization method uses exact linear searches along direction $\mathbf{s}_k = -\mathbf{D}_k\mathbf{g}_k$, where $\mathbf{D}_k$ is assumed to be an arbitrary positive definite matrix and $\mathbf{g}_k$ is the gradient of (5) evaluated at $\mathbf{x}_k$. For any starting point, the convergence rate of such an algorithm could be bounded by the inequity (ref. 6)

$$\frac{F(\mathbf{x}_{k+1}) - F(\mathbf{x}^*)}{F(\mathbf{x}_k) - F(\mathbf{x}^*)} \leq \left(\frac{\kappa(\mathbf{M}_k) - 1}{\kappa(\mathbf{M}_k) + 1}\right)^2 \tag{6}$$

where $\kappa(\mathbf{M}_k) \geq 1$ is the condition number defined as the ratio of the largest eigenvalue of $\mathbf{M}_k$ to the smallest one. The matrix $\mathbf{M}_k$ is given by the formula

$$\mathbf{M}_k = \mathbf{H}^{1/2}\mathbf{D}_k\mathbf{H}^{1/2} \tag{7}$$

Clearly, the fastest convergence can be obtained for Newton type algorithms when $\mathbf{D}_k = \mathbf{H}^{-1}$. Then $\kappa(\mathbf{M}_k) = \kappa(\mathbf{I}) = 1$ and the minimum is reached in only one step. The performance of the simple steepest descent method with exact line searches ($\mathbf{D}_k = \mathbf{I}$) depends heavily on the type of the objective function characterized by the configuration of $\mathbf{H}$ eigenvalues ($\kappa(\mathbf{M}_k) = \kappa(\mathbf{H})$). When these eigenvalues are significantly different, $F(\mathbf{x})$ forms a narrow "valley" and we may anticipate a poor convergence rate. Other search strategies, such as traditional variable metric methods, can increase (or decrease) the convergence rate through the $\mathbf{D}_k$ matrix. It is interesting to note, that for badly selected $\mathbf{D}_k$ it is possible that $\kappa(\mathbf{M}_k) >> \kappa(\mathbf{H})$ and our abstract optimization algorithm may perform worse than the steepest descent method.

Figure 1 illustrates changes in location of $\mathbf{M}_k$ eigenvalues after subtracting $(\mathbf{D}_k\Delta\mathbf{g}_k)(\mathbf{D}_k\Delta\mathbf{g}_k)^T/a_k$ in (4) then adding $\rho\Delta\mathbf{w}_k\Delta\mathbf{w}_k^T/b_k$ and $a_k\mathbf{r}_k\mathbf{r}_k^T$ terms, respectively. All initial eigenvalues μ_i (i =1,...,5) are either smaller or larger than $\rho = 2$ in the example. It is easy to see from figure 1 that each extended BFGS update (12) tends to move all eigenvalues of $\mathbf{M}_k$ but the smallest one closer to each other. The parameter ρ determines the value of smallest eigenvalue. Figure 1 shows that when the ρ scalar is constant, it behaves as an "attractor" for other eigenvalues that will tend monotonically (in a weak sense) to ρ in the subsequent iterations. In DFP or BFGS formulae $\rho = 1$; if this setting increases the condition number $\kappa(\mathbf{M}_k)$, then the convergence of the training method may suffer. One can easily construct examples when all eigenvalues of the initial matrix $\mathbf{M}_0 = \mathbf{H}^{1/2}\mathbf{D}_0\mathbf{H}^{1/2} = \mathbf{H}$ (assuming $\mathbf{D}_0 = \mathbf{I}$) are located away from one. Then, assigning $\rho = 1$ in update (4) may cause convergence deterioration. It may then take a considerable number of iterations for the $\mathbf{M}_k$ eigenvalues to gravitate to each other, so $\kappa(\mathbf{M}_k)$ will decrease and the optimization algorithm will recover its speed.

4. Scaling Factor

In practice, the gain in performance from the correct scaling factor ρ_k is accumulated throughout several iterations. This makes it difficult to adjust ρ_k in an on-line fashion by observing the effects of the error reduction in a short horizon. A proper setting of the ρ_k parameter in (4), so the condition number $\kappa(\mathbf{M}_k)$ will not increase, requires certain information about $\mathbf{M}_k$ eigenvalues. Since the elements of $\mathbf{M}_k$ are not known, similarly to (ref. 6) we use the Rayleigh quotient R(·) to estimate the spread of eigenvalues. For a real and

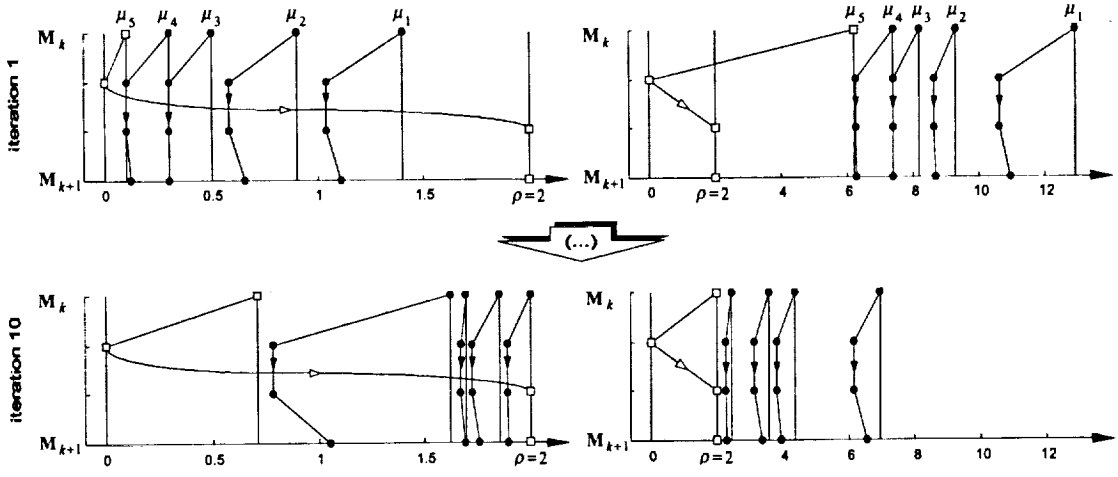


Figure 1. Plots tracking eigenvalue changes of the matrix $\mathbf{M}_k$ due to the repeatedly applied update (4). The "movement" of the smallest eigenvalue of $\mathbf{M}_k$ is marked with the white squares.

symmetric matrix $\mathbf{M}_k$, the Rayleigh quotient is defined by $R_0(\mathbf{x}) = \mathbf{x}^T\mathbf{M}_k\mathbf{x}/\mathbf{x}^T\mathbf{x}$. The lower bound for the largest eigenvalue of $\mathbf{M}_k$ is given by

$$\max(|\mu_i|) \geq R_0(\mathbf{x}), \quad \mathbf{x} \neq 0 \tag{8}$$

Similarly, we may try to estimate the upper bound of the smallest eigenvalue of $\mathbf{M}_k$ using $R_1(\mathbf{x}) = \mathbf{x}^T\mathbf{M}_k^{-1}\mathbf{x}/\mathbf{x}^T\mathbf{x}$

$$\min(|\mu_i|) \leq R_1^{-1}(\mathbf{x}), \quad \mathbf{x} \neq 0 \tag{9}$$

Since $\mathbf{D}_k$ is positive definite and $\mathbf{M}_k = \mathbf{H}^{1/2}\mathbf{D}_k\mathbf{H}^{1/2}$, therefore, $\mu_i \geq 0$. This allows us to omit the absolute value operators in (8) and (9). $R_0(\mathbf{x})$ and $R_1(\mathbf{x})$ are equal to the largest and smallest eigenvalues respectively, when $\mathbf{x}$ is the associated eigenvector; in other cases these estimators are less accurate. For certain vectors $\mathbf{x}$ the Rayleigh quotients can be computed rather inexpensively

$$R_0\left(\mathbf{H}^{1/2}\Delta\mathbf{w}_k\right) = \frac{\Delta\mathbf{g}_k^T\mathbf{D}_k\Delta\mathbf{g}_k}{\Delta\mathbf{w}_k^T\Delta\mathbf{g}_k} = \frac{a_k}{b_k} \tag{10}$$

$$R_1^{-1}\left(\mathbf{H}^{-1/2}\Delta\mathbf{g}_k\right) = \frac{\Delta\mathbf{g}_k^T\Delta\mathbf{w}_k}{\Delta\mathbf{w}_k^T\mathbf{D}_k^{-1}\Delta\mathbf{w}_k} = \frac{b_k}{c_k} \tag{11}$$

The scalar $c_k = \Delta\mathbf{w}_k^T\mathbf{D}_k^{-1}\Delta\mathbf{w}_k$ in (11) can be calculated without inverting $\mathbf{D}_k$. Taking advantage of the relationship $\Delta\mathbf{w}_k = -\alpha_k\mathbf{D}_k\mathbf{g}_k$ used to compute weight corrections (section 5) we can write

$$c_k = \Delta\mathbf{w}_k^T\mathbf{D}_k^{-1}\Delta\mathbf{w}_k = -\alpha_k\Delta\mathbf{w}_k^T\mathbf{g}_k \tag{12}$$

In our training algorithm ρ_k is adjusted based on the geometric average of (29) and (30), i.e.

$$\rho_k = \sqrt{R_0\left(\mathbf{H}^{1/2}\Delta\mathbf{w}_k\right)R_1^{-1}\left(\mathbf{H}^{-1/2}\Delta\mathbf{g}_k\right)} = \sqrt{a_k/c_k} \tag{13}$$

Unfortunately, in some optimization problems estimator (13) tends to vary rapidly. Many fluctuations of ρ_k may be associated with the imperfect procedure of evaluating this parameter rather than the real changes of the eigenvalues and the error surface. In section 2 we mentioned possible benefits of slowly varying ρ_k by pointing out that for constant $\rho_k = \rho$ the convergence $\mathbf{D}_k \to \rho\mathbf{H}^{-1}$ is achieved for the sequence of matrices generated by the updating formula (4). In the SVM algorithm, we suppress variations of $1/\rho_k$ rather than ρ_k by implementing a simple, low-pass filter using an exponential average

$$\rho_k^{-1} = \gamma\rho_{k-1}^{-1} + (1-\gamma)\tau\sqrt{c_k/a_k} \tag{14}$$

where γ is the weighting factor ($0 \leq \gamma < 1$, typically, $\gamma = 0.9$). The τ scalar in (14) has heuristic roots and is used to drive ρ_k slightly toward smaller values, which have typically better convergence properties; in the SVM algorithm we set $\tau = 1.2$.

Figure 2 displays changes of the scaling factor during the training procedure. Apparently, in this case the rescaling of the $\mathbf{D}_k$ matrix is not necessary during the very first stages of training. However, as the training progresses other values of ρ_k, different from 1.0, are more appropriate.

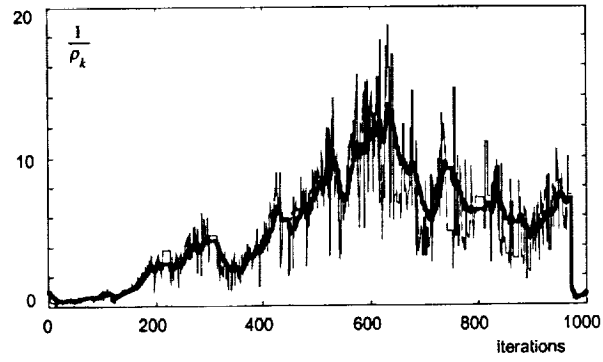


Figure 2. Changes of the scaling parameter $1/\rho_k$ during training evaluated according to equation (14). Both filtered values and raw estimates $\tau\sqrt{c_k/a_k}$ are displayed.

5. Step Length Calculation

In the variable metric methods, the search direction is determined from the equation $\mathbf{s}_k = -\mathbf{D}_k\mathbf{g}_k$. Then, the step length α_k and the weight correction $\Delta\mathbf{w}_k = \alpha_k\mathbf{s}_k$ along a given ray are established. To avoid expensive direct line searches, we compute α_k according to the formula (refs. 7 & 8)

$$\alpha_k = -\frac{\mathbf{s}_k^T\mathbf{g}_k}{\mathbf{s}_k^T\mathbf{H}\mathbf{s}_k + \lambda\mathbf{s}_k^T\mathbf{s}_k} \tag{15}$$

for which the vector $\mathbf{H}\mathbf{s}_k$ is evaluated using the RBackprop algorithm (ref. 8). To achieve a more precise

step size adjustment, the λ parameter is continuously tuned during the training process. This process is performed differently from (ref. 7).

For a descent search direction and sufficiently large value of λ, it should always be possible to find such an α_k that (15) would lead to the error minimization. However, this primary condition ($E(\mathbf{w}_{k+1}) < E(\mathbf{w}_k)$) for the step size to be accepted does not assure an efficient training strategy. Under certain circumstances, the step size defined by (15) may be either too small so the new point is placed far before the minimum or too large; in the latter case the algorithm overshoots the minimum along the given ray producing negligible error reduction. In both situations, the result is an unsatisfactory decrease of the objective function. For positive curvature ($\mathbf{s}_k^T\mathbf{H}\mathbf{s}_k > 0$) and $E(\mathbf{w})$ represented along the search direction by the function of one argument $E(\alpha_k) = E(\mathbf{w}_k + \alpha_k\mathbf{s}_k)$ a simple criterion against too long step size is (refs. 4 & 9)

$$E(\alpha_k) \le E(0) + \beta\alpha_k E'(0) = E(0) + \beta\alpha_k\mathbf{g}_k^T\mathbf{s}_k \qquad (16)$$

where β ($0 \le \beta \le 0.5$) is a fixed parameter. Setting β to the values smaller than 0.5 relaxes condition (16) allowing longer steps to be taken. In our training algorithm we use $\beta = 0.5$.

The SVM algorithm attempts to take full advantage of the presumed local quadratic nature of $E(\alpha_k)$ by reducing λ as much as possible. The value of λ is decreased by half on every iteration that satisfies (16). When the condition is violated but the reduction in error is achieved regardless, the new weight settings are accepted and the value for λ is increased (multiplied by two) so in subsequent iterations the step length α_k will tend to be smaller. Finally, in case of failure ($E(\mathbf{w}_{k+1}) \ge E(\mathbf{w}_k)$), the value of λ is increased more rapidly (multiplied by four) resulting in shorter step sizes α_k in the next attempts to minimize the error function.

In the special case when the denominator in (15) is not positive, λ is reset to the new value

$$\lambda \leftarrow \lambda + 2\frac{|\mathbf{s}_k^T\mathbf{H}\mathbf{s}_k|}{\mathbf{s}_k^T\mathbf{s}_k} \qquad (17)$$

This rather "desperate" act effectively reverses the Hessian sign in the step length calculations. Resetting λ according to (17) acts as a safeguard in rare situations when the local information $\mathbf{H}\mathbf{s}_k$ and the current value of λ does not provide reliable information on how much the step size could be extended.

In our algorithm, the λ parameter is bounded between *macheps*[1] and 10^{16} for a double precision implementation. Violation of the upper constraint was chosen to signal the failure to minimize $E(\mathbf{w})$. It may be possible that $\mathbf{D}_k$ is no longer positive definite due to the round-off errors and the search direction is not a descent one. In such a case, $\mathbf{D}_k$ is reset to the identity matrix and consequently, the new search direction $\mathbf{s}_k = -\mathbf{g}_k$ is selected.

Note that the training method described here does not automatically reset $\mathbf{D}_k$ to the identity matrix every N iterations as in the cyclic methods (N - total number of weights). We argue that, for large optimization problems, discarding previously acquired information about the second derivatives in such predominant fashion is often unnecessary. In our method, the $\mathbf{D}_k$ matrix is reset when it is evident that its positive definite property is lost, i.e., $a_k \le 0$. In addition, another test is performed

$$\frac{b_k}{\|\Delta\mathbf{w}_k\|_2\|\Delta\mathbf{g}_k\|_2} > macheps \qquad (18)$$

to avoid updating $\mathbf{D}_k$ with two noisy vectors, $\Delta\mathbf{w}_k$ and $\Delta\mathbf{g}_k$. Note that (18) assures $b_k > 0$. If condition (16) is false, the $\mathbf{D}_k$ update should be skipped. Furthermore, when this situation is detected in several consecutive iterations the Hessian approximation should be reset as well.

6. Predictive Updating

In the basic quasi-Newton schemes, matrix $\mathbf{D}_k$ is always updated after the actual step from $\mathbf{w}_k$ to $\mathbf{w}_{k+1}$ is made. Therefore, the choice of the search direction relies only on previously acquired information, without precise knowledge of what could be expected in the next step. The RBackprop algorithm could be used to partially compensate for this deficiency by probing desired directions.

---

[1] For floating point computations, the machine precision (*macheps*) may be defined as the smallest value so 1.0 and 1.0+*macheps* have different representations in the computer memory.

Table 1. Final training errors and average CPU time used to complete a single training task.

Method / Run	SVM	LM	BFGS	BFGS limited memory	RPROP
1	3.968e-03	4.007e-03	5.325e-03	5.102e-03	6.606e-03
2	4.248e-03	4.146e-03	4.707e-03	5.372e-03	6.691e-03
3	4.201e-03	4.286e-03	4.975e-03	5.347e-03	6.790e-03
4	4.299e-03	4.423e-03	5.351e-03	5.706e-03	6.702e-03
5	4.180e-03	3.987e-03	5.051e-03	5.157e-03	6.644e-03
6	4.065e-03	4.209e-03	5.170e-03	5.220e-03	7.147e-03
7	4.126e-03	3.991e-03	5.015e-03	5.297e-03	6.218e-03
8	3.906e-03	3.809e-03	5.123e-03	5.376e-03	6.670e-03
9	3.762e-03	3.992e-03	5.053e-03	5.391e-03	6.802e-03
10	4.389e-03	4.477e-03	5.269e-03	5.428e-03	6.667e-03
Iterations	1000	200	1000	1000	1000
CPU time	25 min	14 h 40 min	26 min	22 min	5 min

The idea of predictive updating is not complicated. A search direction that is computed as in the traditional variable metric algorithms serves as the first approximation of the final search ray. Along this direction, information about second derivatives is collected using the RBackprop method. This information is used to update the $\mathbf{D}_k$ matrix one more time before calculating the final search direction.

The initial guess for the search direction is therefore $\mathbf{s}_k = -\mathbf{D}_k\mathbf{g}_k$. In a small neighborhood of the current point $\mathbf{w}_k$, the error function can be approximated by a quadratic model

$$E(\mathbf{w}_k) \approx c + \mathbf{b}^T\mathbf{w}_k + \tfrac{1}{2}\mathbf{w}_k^T\mathbf{H}\mathbf{w}_k \qquad (19)$$

where $\mathbf{b}$ is such a vector that $\partial E(\mathbf{w}_k)/\partial \mathbf{w} = \mathbf{b}+\mathbf{H}\mathbf{w}_k$ and c is constant. Operating on (19), for some suitable ε and $\Delta\mathbf{w} = \varepsilon\mathbf{s}_k$ we can write

$$\left.\frac{\partial E}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}_k+\varepsilon\mathbf{s}_k} - \left.\frac{\partial E}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}_k} = \Delta\mathbf{g} \approx \varepsilon\mathbf{H}\mathbf{s}_k \qquad (20)$$

Relationship (20) suggests that the $\mathbf{D}_k$ matrix may be updated using $\varepsilon\mathbf{H}\mathbf{s}_k$ and $\varepsilon\mathbf{s}_k$ in place of $\Delta\mathbf{g}$ and $\Delta\mathbf{w}$ in the equation defining the extended BFGS formula. Note that the ε scalar will annihilate in this update. This yields the new matrix $\breve{\mathbf{D}}_k$, which can used to compute the final search direction $\breve{\mathbf{s}}_k = -\breve{\mathbf{D}}_k\mathbf{g}_k$.

Employing a supplementary, predictive update of the $\mathbf{D}_k$ matrix involves some risks when the error function is highly non-quadratic. Our tests showed, however, a positive impact of the predictive update in practical problems. In the majority of tests, the convergence was faster and the final results were better in comparison to the method that used a single BFGS update per epoch.

Computer implementation of the predicative updating is not complicated, as the additional program code is very similar to the main loop in extended BFGS routine. Moreover, since the network output and the gradient were already evaluated for $\mathbf{w}_k$, the product $\mathbf{H}\mathbf{s}_k$ may be obtained relatively inexpensively for any $\mathbf{s}_k$ (at the cost of one additional feedforward and one backward pass) using the RBackprop algorithm.

7. Efficient Update Implementation

It is not unusual for feedforward neural networks to incorporate several hundred adjustable parameters. For variable metric methods, this translates into increased costs for periodic updating of the $\mathbf{D}_k$ matrix. In such cases it may be worthwhile to convert (4) into a formula that clearly looks like a rank two, symmetric update

$$\mathbf{D}_{k+1} = \mathbf{D}_k + \mathbf{u}_k\mathbf{u}_k^T - \mathbf{v}_k\mathbf{v}_k^T \qquad (21)$$

and utilize it in the updating algorithm. In equation (21) vectors $\mathbf{v}_k$ and $\mathbf{u}_k$ are expressed by

$$\mathbf{v}_k = \frac{\mathbf{D}_k\Delta\mathbf{g}_k}{\sqrt{a_k + \rho_k b_k}}, \quad \mathbf{u}_k = \frac{\sqrt{a_k + \rho_k b_k}}{b_k}\Delta\mathbf{w}_k - \mathbf{v}_k \qquad (22)$$

Obviously, when updating the $\mathbf{D}_k$ matrix, its symmetric property should also be exploited to avoid redundant calculations with respect to either lower or upper triangle part. Below, pseudo code is presented for executing the

extended BFGS variable metric update. Using this scheme requires three times less multiplications in the main routine loop in comparison to the procedure which implements (4) directly.

```
v = D * Δg;
a = Δg^T * v;
b = Δw^T * Δg;
if (a > 0.0 AND
    b > macheps * (Δw^T * Δw) * (Δg^T * Δg))
{
  a = sqrt(a + ro * b);
  v = v / a;
  u = a / b * Δw - v;
  for (i = 0; i < n; i++)
  {
    a = u[i];
    b = v[i];
    D[i][i] += a * a - b * b;
    for (j = i + 1; j < n; j++)
    {
      D[j][i] = (D[i][j] += a*u[j]-b*v[j]);
    }
  }
}
```

8. Numerical Experiments

Numerical experiments have been carried out to test the performance of the scaled variable metric method against other selected training techniques which are known to be effective and frequently used in practice. The following four training algorithms were chosen for the comparison: (i) Levenberg-Marquardt (LM) algorithm with the predicted error reduction (ref. 9), (ii) standard BFGS method (ref. 2), (iii) limited memory BFGS (ref. 9), and (iv) RPROP (ref. 10). Both BFGS methods employed Brent's line search (ref. 11). A common updating routine of the $\mathbf{D}_k$ matrix was implemented in the identical fashion in the SVM method as in other variable metric algorithms. All numerical tests were performed on a Pentium 200 MHz personal computer.

The performance comparison of the SVM method was experimentally verified by training ten, 12-18-16-6 feedforward neural networks (12 input sensors, 6 outputs, 640 weights) using 1373 preprocessed data points acquired from the calibration process of a six-component wind tunnel strain-gage. Each of the ten training experiments used the same starting point for all the algorithms. Figure 3 presents the convergence curves of the five training algorithms. Clearly, the simplest RPROP algorithm exhibited the lowest convergence rate. The standard BFGS method and its limited memory version demonstrated a better performance but the SVM algorithm was superior to these techniques. On average, our algorithm was able to reach the error level of the standard BFGS method in 250-350 iterations. The steepest convergence curve belonged to the Levenberg-Marquardt algorithm. However, since the neural model had multiple outputs, the Levenberg-Marquardt training was substantially slower than other methods, requiring 14 hours and 40 minutes to iterate 200 epochs. The SVM method (run for 1000 epochs) was able to surpass the Levenberg-Marquardt results in most cases, requiring only 25 minutes on average to complete training. The RPROP method was the fastest training technique in our comparison. It is important to note, however, that for the given problem, the RPROP algorithm was least accurate of all the methods under consideration, even when the number of iterations was increased to 5000. In this case, execution time was approximately 25 minutes.

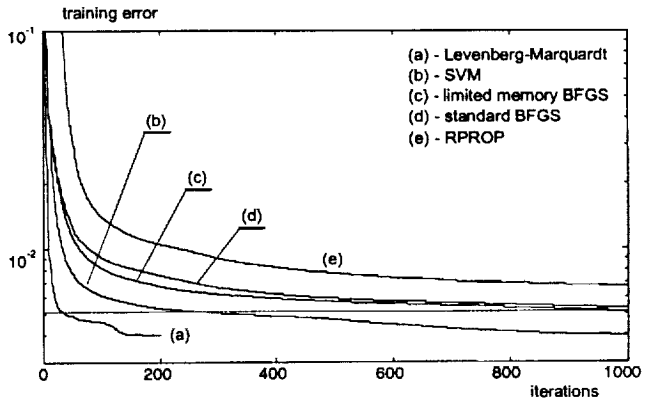


Figure 3. Convergence curves of different training algorithms.

9. Conclusions

In this report, we have presented a new scaled variable metric (SVM) method for training feedforward neural networks. The SVM technique utilizes the RBackprop algorithm (ref. 8) and the modified variable metric update, derived as a subclass of the Huang family formulae. The variable metric method is used to collect information about second derivatives of the error function with respect to network weights. It allows us to construct a relatively efficient strategy for choosing the sequence of search directions. The variable metric update was extended by an additional parameter ρ_k, which plays a fundamental role in attempts to accelerate the convergence speed of the training procedure. We have shown that a special case of the Huang updating formulae can be efficiently combined with the

variable metric algorithm (ref. 6) to form an efficient training scheme. We have presented a new strategy for setting the scaling factor, which emphasizes the importance of limiting unnecessary fluctuations of ρ_k.

The RBackprop algorithm can be utilized in two ways: it allows us to avoid expensive one directional line searches, and it can be used in supplementary, predictive updating of the $\mathbf{D}_k$ matrix. Predictive updating acquires information about second derivatives along the trial search direction before committing to the final step which can be chosen more precisely. In addition, matrix $\mathbf{D}_k$ may benefit from the predictive updates by being able to track changes of the Hessian matrix faster. We have employed a modified strategy for adjusting the λ parameter in the step length calculations using directional slope testing. We have found that this technique works exceptionally well in conjunction with the RBackprop algorithm. Finally, we have outlined a computationally more efficient scheme for updating the $\mathbf{D}_k$ matrix. This is a somewhat overlooked aspect of many variable metric implementations but it becomes a rather important issue when large neural networks are being trained.

Numerical experiments provide evidence that the theoretical background developed to estimate an appropriate range of settings for the ρ_k scalar indeed works in practice, although the same theory indicates that acceleration may not always be possible. For some problems, where $\rho_k \approx 1$ is a suitable choice, the standard BFGS method may be superior. In practical situations, however, the SVM algorithm may protect the variable metric algorithm from being stuck. The method also has the capacity to perform a continuous adjustment of the ρ_k parameter rather than rescale the initial matrix $\mathbf{D}_0$ only once. Interestingly, in some cases the initial scaling of the $\mathbf{D}_0$ matrix is not necessary, but later adjustment of the ρ_k parameter improves the training convergence.

It seems that information provided by the RBackprop algorithm and inferred from gradient vectors may be more efficiently utilized in choosing the sequence of search directions when the ρ_k scalar is allowed to be tuned on-line. Our experience with the SVM method is that its efficiency becomes evident when the size of the neural architecture increases and/or the difficulty achieving low error arises, perhaps due to the highly non-quadratic nature of the optimization problem.

References

1. Battiti, R., First- and second-order methods for learning: between steepest descent and Newton's method, *Neural Computation*, 4, 1992, pp. 141-166.

2. Bishop, C.M., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

3. Luksan, L., Computational experience with known variable metric updates, *Journal of Optimization Theory and Applications*, Vol. 83, No. 1, 1994, pp. 27-47.

4. Wolfe, M.A., *Numerical Methods for Unconstrained Optimization*, Van Nostrad Reinhold, 1978.

5. Huang, H.Y., Unified approach to quadratically convergent algorithms for function minimization, *Journal of Optimization Theory and Applications*, Vol. 5, No. 6, 1970, pp. 405-423.

6. Oren, S.S., and Luenberger, D.G., Self-scaling variable metric (SSVM) algorithms, Part I and II, *Management Science*, Vol. 20, No. 5, 1974, pp. 845-874.

7. Moller, M.F., A scaled conjugate gradient algorithm for fast supervised learning, *Neural Networks* Vol. 66, No. 4, 1993, pp. 525-533.

8. Pearlmutter, B.A., Fast exact multiplication by the Hessian, *Neural Computation*, 6 (1), 1994, pp. 147-160.

9. Fletcher, R., *Practical Methods of Optimization*, John Wiley & Sons, 1987.

10. Riedmiller, M., and Braun, H., A direct adaptive method for faster backpropagation learning: the RPROP algorithm, *IEEE Int. Conf. Neural Networks*, San Francisco, 1993, pp. 586-591.

11. Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P., *Numerical Recipes in C– The Art of Scientific Computing*, 2nd ed., Cambridge University Press, 1992.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 1998	Technical Memorandum

4. TITLE AND SUBTITLE

Accelerated Training for Large Feedforward Neural Networks

5. FUNDING NUMBERS

RTOP 519-30-12

6. AUTHOR(S)

Slawomir W. Stepniewski and Charles C. Jorgensen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Ames Research Center
Moffett Field, CA 94035-1000

8. PERFORMING ORGANIZATION REPORT NUMBER

A9812323

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA/TM-1998-112239

11. SUPPLEMENTARY NOTES

Point of Contact: Charles Jorgensen, Ames Research Center, 269-1, Moffett Field, CA 94035-1000
(650) 604-6725

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified-Unlimited
Subject Category – 59

Availability: NASA CASI (301) 621-0390

12b. DISTRIBUTION CODE

Distribution: Standard

13. ABSTRACT ***(Maximum 200 words)***

In this paper we introduce a new training algorithm, the scaled variable metric (SVM) method. Our approach attempts to increase the convergence rate of the modified variable metric method. It is also combined with the RBackprop algorithm, which computes the product of the matrix of second derivatives (Hessian) with an arbitrary vector. The RBackprop method allows us to avoid computationally expensive, direct line searches. In addition, it can be utilized in the new, "predictive" updating technique of the inverse Hessian approximation. We have used directional slope testing to adjust the step size and found that this strategy works exceptionally well in conjunction with the Rbackprop algorithm. Some supplementary, but nevertheless important enhancements to the basic training scheme such as improved setting of a scaling factor for the variable metric update and computationally more efficient procedure for updating the inverse Hessian approximation are presented as well. We summarize by comparing the SVM method with four first- and second-order optimization algorithms including a very effective implementation of the Levenberg-Marquardt method. Our tests indicate promising computational speed gains of the new training technique, particularly for large feedforward networks, i.e., for problems where the training process may be the most laborious.

14. SUBJECT TERMS

Neural networks, Training, Optimization, Quasi-Newton methods, Variable metric methods

15. NUMBER OF PAGES

14

16. PRICE CODE

A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified		