Pergamon

Neural Networks, Vol. 7, No. 1, pp. 1–11, 1994

0893-6080/94 $6.00 + .00

INVITED ARTICLE

Minimisation Methods for Training Feedforward Neural Networks

P. PATRICK VAN DER SMAGT

University of Amsterdam

(*Received* 15 *December* 1992; *accepted* 1 *June* 1993)

Abstract—*Minimisation methods for training feedforward networks with back propagation are compared. Feedforward neural network training is a special case of function minimisation, where no explicit model of the data is assumed. Therefore, and due to the high dimensionality of the data, linearisation of the training problem through use of orthogonal basis functions is not desirable. The focus is on function minimisation on any basis. Quasi-Newton and conjugate gradient methods are reviewed, and the latter are shown to be a special case of error back propagation with momentum term. Three feedforward learning problems are tested with five methods. It is shown that, due to the fixed stepsize, standard error back propagation performs well in avoiding local minima. However, by using not only the local gradient but also the second derivative of the error function, a much shorter training time is required. Conjugate gradient with Powell restarts shows to be the superior method.*

Keywords—Feedforward neural network training, Numerical optimisation techniques, Neural function approximation, Error back propagation, Conjugate gradient, Quasi-Newton.

1. INTRODUCTION

Back propagation of error gradients has proven its usefulness in training feedforward and feedback neural networks to tackle a large number of classification and function mapping problems. In many cases, however, the large number of learning iterations needed to optimally adjust the parameters of the networks is prohibitive for online applications, such as adaptive process control. But even for applications where real-time response is not required, the slow convergence of error back propagation often results in training times exceeding hours of computer time.

Numerical analysis has always focused on methods using not only the local gradient of the function but also the second derivatives. In the former case, the function is approximated by the first (constant) and second (linear) terms of a Taylor expansion; in the second case, the third (quadratic) term is also taken into account. When the approximation of the function by its second order Taylor expansion is precise, the global minimum can be found in v iterations, where v is the number of degrees of freedom of the system.

Error functions, however, such as those that are to be minimised in feedforward network training, are not precisely a parabola, but can generally be considered to consist of a summation of parabola, especially close to minima.

Recently, there has been a focus of training feedforward neural networks with conjugate gradient methods (Battiti, 1992; Kinsella, 1992; Barnard, 1992; van der Smagt & Kröse, 1991). Their superiority in solving nonlinear optimisation techniques requires more extensive use in back propagation learning, while a better understanding of getting trapped in local minima, and their behaviour not close to global minima, is imperative.

Acknowledgements: This work has been sponsored by the Dutch Foundation for Neural Networks. The author expresses his gratitude to the Carver Charitable Trust for support of his work at the Beckman Institute, University of Illinois. Computer simulations were performed on machines of the Theoretical Biophysics group, and on NCSA machines, both at the Beckman Institute, University of Illinois, Urbana, IL 61801; the author is grateful for their generosity. Finally, the author wishes to thank Prof. Dr. F. C. A. Groen, Dr. W. Hoffmann, and Prof. Dr. K. S. Schulten for useful discussions.

Requests for reprints should be sent to the author at the Department of Computer Science and Mathematics, University of Amsterdam, Kruislaan 403, NL-1098 SJ Amsterdam, The Netherlands.

2. APPROXIMATION OF AN UNKNOWN FUNCTION

We consider a function $\mathcal{F} : \Re^m \rightarrow \Re^n$ that is to be approximated. The function $\mathcal{F}$ is not analytically known,

but rather samples $S^P = \{s^1, s^2, \ldots, s^P\}$ with $s^p = (\mathbf{x}^p, \mathbf{y}^p)$ are generated by a process that is governed by $\mathcal{F}$, that is, $\mathcal{F}(\mathbf{x}^p) = \mathbf{y}^p$. From the available samples we want to build a smooth approximation of $\mathcal{F}$.

To approximate $\mathcal{F}$ we regard a feedforward neural network $\mathcal{N}_S^{\mathbf{w}} : \Re^m \to \Re^n$. $\mathcal{N}_S^{\mathbf{w}}$ consists of *basis functions* $\phi_i : \Re \to \Re$, $(1 \le i \le \kappa)$ that are represented by hidden units, and weighted connections between the input, hidden, and output units. The ith output element of $\mathcal{N}$ is

$$\mathcal{N}_S^{\mathbf{w}}(\mathbf{x})_i = \sum_{j=1}^{\kappa} w_{2ij}\phi_j\left(\sum_{k=1}^{m} w_{1jk}x_k + w_{1,jk+1}\right) + w_{2,i,j+1}, \qquad 1 \le i \le n, \quad (1)$$

with one layer of hidden units and linear output units.

Without loss of generality, we assume $\mathcal{N}$ to have a fixed (i.e., predetermined) topology. The approximation $\mathcal{N}_S^{\mathbf{w}}$ then only depends on the learning samples S, and the learning algorithm that determines the parameters $\mathbf{w}$ from S and the architecture of $\mathcal{N}$. In this paper, an overview of methods to heuristically determine optimal parameters $\mathbf{w}$ that use not only first but also second derivatives of $\mathcal{F}$ are reviewed.

When approximating $\mathcal{F}$ with $\mathcal{N}$, we have to consider three types of error: representation error, generalisation error, and optimisation error (Vyšniauskas, Groen, & Kröse, 1992).

Representation Error. Let us first consider the case when the full set of learning samples S^∞ is available[1]. Also, assume that, given S^∞, we can find the optimal $\mathbf{w}^{\text{opt}}$. On a digital computer, this can theoretically be obtained via a brute-force search through the whole (finite but very large) parameter space. This will render $\mathcal{N}_\infty^{\text{opt}}$, such that

$$\int \|\mathcal{F}(\mathbf{x}) - \mathcal{N}_\infty^{\text{opt}}(\mathbf{x})\|\, d\mathbf{x} < \varepsilon_r \quad (2)$$

is minimal and only depends on the architecture (c.q., number of hidden units) of $\mathcal{N}$. Equation (2) is called the *representation error* (Hornik, Stinchcombe, & White, 1989).

Generalisation Error. In real-world applications, only a finite (i.e., small) number of learning samples is available or can be used at the same time. Furthermore, the samples contain noise. The values of $\mathcal{F}$ for which no samples are available must be interpolated. Thus, a generalisation error occurs, consisting of the errors for those pairs $(\mathbf{x}, \mathbf{y}) \notin S$. The generalisation error is included in eqn (2):

$$\int \|\mathcal{F}(\mathbf{x}) - \mathcal{N}_S^{\text{opt}}(\mathbf{x})\|\, d\mathbf{x} < \varepsilon_r + \varepsilon_g. \quad (3)$$

Optimisation Error. Because there is only a finite set of learning samples available, evaluation of eqn (3) is not feasible; the approximation is evaluated only at the values $\mathbf{x}$ where there are available learning samples. The approximated error that is used is

$$E = \sum_{(\mathbf{x},\mathbf{y})\in S} \|\mathbf{y} - \mathcal{N}_S^{\mathbf{w}}(\mathbf{x})\| < \varepsilon_r + \varepsilon_g + \varepsilon_o. \quad (4)$$

Equation (4) is the function that is minimised in the learning process. The function $\|\cdot\|$ is generally interpreted as the L_2 norm, that is, $\|\zeta\|_2 = \sqrt{\sum_i \zeta_i^2}$. In some cases, the L_∞ norm (or *max* norm) is suggested, especially as a criterion for stopping the minimisation process; however, in the presence of noise (that is, in the case of real noise, Poisson distributed) the L_∞ norm is not meaningful.

This paper concentrates on the error ε_o obtained during the minimisation process.

2.1. Nonlinear Optimisation Techniques

The problem of optimal representation is thus reduced to first determining the structure (c.q., choice of basis functions $\phi_1, \phi_2, \ldots, \phi_\kappa$) of the network, and from that finding an optimal set $\mathbf{w}$, where optimality is interpreted in terms of eqn (4).

Obviously, these two processes are intertwined. If a good set of basis functions can be found, the success of which depends on the particular problem, then the second step may become easier to perform.

Incremental Learning. Although error back propagation is best used on a batch of learning samples, when those samples are generated one at a time while the learning process is in progress it is desirable to update the approximation with those new samples. Because more than the local gradient is necessary, second-order methods cannot directly be used with sample training. The following modification may present a solution, however. Instead of eqn (4) we measure the change in the approximation of the neural network over the whole input domain from weight vector $\mathbf{w}_i$ to $\mathbf{w}_{i+1}$, and try to minimise it. This *minimum entropy method* is combined in a weighted sum with the error caused by the new learning sample (see Figure 1).

Thus, the error function that has to be minimised is

$$\int \|\mathcal{N}_S^{\mathbf{w}^{i+1}}(\mathbf{x}) - \mathcal{N}_S^{\mathbf{w}^i}(\mathbf{x})\|\, d\mathbf{x}. \quad (5)$$

This minimum entropy method, suggested by Kadirkamanathan & Fallside (1990), works well when ac-

[1] The limited representation accuracy of a digital computer makes it unnecessary, indeed, that S has infinitely many samples.

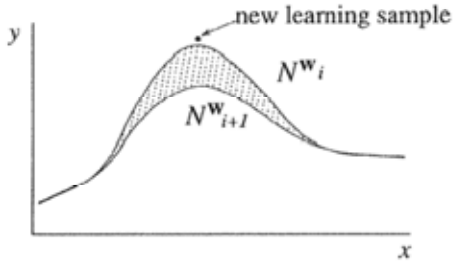

FIGURE 1. Although the new learning sample has to be approximated, the difference between the previous and the new approximation (shaded area) is minimised.

tivation functions are used that only have a local influence on the approximation; otherwise, evaluating eqn (5) is too costly.

3. CHOOSING THE BASIS FUNCTIONS

Choosing a good set of basis functions ϕ_i reduces the burden of finding a good set of parameters $\mathbf{w}_i$. In particular, when the error function eqn (4) is linear in the parameters $\mathbf{w}$, the optimal solution can be analytically found[2].

Such linear dependency can be obtained by intelligently choosing your basis functions. An obvious choice often is to use basis functions $\{x^i\}$, but orthogonal polynomials are preferred. Appendix A discusses the advantage of orthogonal polynomials over nonorthogonal ones.

3.1. Orthogonal Basis Functions

Orthogonality of two functions $\phi_i(\mathbf{x})$ and $\phi_j(\mathbf{x})$ is defined as

$$\int_{\mathbf{x}_a}^{\mathbf{x}_b} \phi_i(\mathbf{x})\phi_j(\mathbf{x})\, d\mathbf{x} = c\delta_{ij} \tag{6}$$

over some interval $[\mathbf{x}_a, \mathbf{x}_b]$, where δ_{ij} is the Kronecker delta, and c is a constant. When $c = 1$, the functions are called *orthonormal.*

Because we have a finite set of samples, the orthonormality of eqn (6) is interpreted only in terms of S, that is, it is required that

$$\sum_{(\mathbf{x},\mathbf{y})\in S} \phi_i(\mathbf{x})\phi_j(\mathbf{x}) = \delta_{ij}. \tag{7}$$

Assuming a set $\{\phi_i\}$ of orthonormal basis functions can be found, the neural network becomes linear in its parameters, such that eqn (1) can be written as

$$\mathcal{N}_S^{\mathbf{w}}(\mathbf{x})_i = \sum_{j=1}^{\kappa} w_{1ij}\phi_j(x_j), \quad 1 \le i \le n. \tag{8}$$

The optimal network parameters can be determined by minimisation of the error functions

$$E_i = \sum_{(\mathbf{x},\mathbf{y})\in S} \left\| y_i - \sum_{j=1}^{\kappa} w_{1ij}\phi_j(x_j) \right\|, \quad 1 \le i \le n, \tag{9}$$

in parameters w_{1ij}. When $P > \kappa$, eqn (9) can be solved by minimising all of its terms independent of each other.

Finite orthogonality can be acquired by use of a three-term recursive generator (Stoer & Bulirsch, 1980); consult Forsythe (1957) for an elegant implementation. With this method, the orthogonal polynomials are constructed incrementally (i.e., updated for each new sample), such that the system can accommodate new learning samples in $O(\kappa)$ time.

Practice shows that polynomials are rather *stiff* (i.e., not flexible) for fitting data; especially when polynomials of high order are used, the fitted curve tends to oscillate. An alternative method is given by *splines.* Here, the space between two measurements $(\mathbf{x}^p, \mathbf{y}^p)$ and $(\mathbf{x}^{p+1}, \mathbf{y}^{p+1})$ is *locally* interpolated by a single spline (polynomial) by only looking at the values $\mathbf{y}^p$ and $\mathbf{y}^{p+1}$ and their derivatives or, in the case of cubic splines, their second derivatives.

A different type of orthonormality can be obtained with local representations. To represent each learning sample, delta functions can be used:

$$\phi(\mathbf{x}) = \delta(\mathbf{z} - \mathbf{x}) := \begin{cases} 1 & \text{if } \mathbf{z} = \mathbf{x}, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

In effect, a table lookup method is implemented. Clearly, these functions adhere to orthonormality of eqn (7), and learning is a trivial task. The approximation is local, and smoothness has to be added separately using some interpolation technique. For neural implementations of this and similar techniques see Miller III (1989), and van der Smagt, Jansen, and Groen (1992).

For functions $\mathcal{F}$ of more than one variable, constructing orthonormal interpolating basis functions or finding splines becomes cumbersome and even prohibitive when the data samples are not arranged on a grid. For problems of high dimensionality, orthonormal basis functions are impractical in use.

3.2. Nonorthonormal Basis Functions

We therefore revert to basis functions that are not orthonormal, and are quantitatively independent of the available data samples. Again, eqn (1) is the representation of the neural approximator, and the error functions that must be minimised are

[2] Roundoff errors might require that the optimal solution be computed twice or thrice, instead of only once, which would theoretically suffice to find the $\mathbf{w}^{\text{opt}}$.

$$E_i = \sum_{(\mathbf{x},\mathbf{y})\in S} \left\| y_i - \sum_{j=1}^{s} w_{2ij}\phi_j\left(\sum_{k=1}^{m} w_{1jk}x_k + w_{1,jk+1}\right) + w_{2,j,j+1}\right\|, \quad 1 \le i \le n. \quad (11)$$

Because no model of $\mathcal{F}$ is assumed, the basis functions are chosen on basis of their flexibility and cost. A common choice is the sigmoid function $\sigma(\cdot)$ because of its nice properties.

4. PARAMETER ESTIMATION

Having determined a set of basis functions $\phi_i(\cdot)$, an optimal set of parameters $\mathbf{w}$ have to be found to minimise the summed squared error

$$E(\mathbf{w}) = \sum_{i=1}^{n} \sum_{(\mathbf{x},\mathbf{y})\in S} \tfrac{1}{2}\left[y_i - \sum_{j=1}^{s} w_{2ij}\phi_j\left(\sum_{k=1}^{m} w_{1jk}x_k + w_{1,jk+1}\right) + w_{2,i,j+1}\right]^2. \quad (12)$$

This learning process can be depicted as a walk through a very high-dimensional weight space (see Figure 2).

Even though an optimal weight vector (point A) exists for a given network structure, the limited learning set will render a different optimal solution (point B) within the set W^{opt} of optimal solutions. Due to imperfect minimisation a suboptimal solution (point C) is reached. Also note that, due to the possibility of permutations in the weights and hidden units (Chen & Hecht–Nielsen, 1989), the solutions for $\mathbf{w}^{\text{opt}}$ are not unique, but the weight space consists of a number of identical cones, each containing a global optimum.

Iterative minimisation methods are based on the following principle: given are a function $E(\mathbf{w})$ that is to be minimised, and an initial state $\mathbf{w}_0$. Now for each iteration, find a minimisation *direction* $\mathbf{u}_i$ and a *stepsize* α_i, and update $\mathbf{w}$ as

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \alpha_i\mathbf{u}_i. \quad (13)$$

The task of minimisation is finding optimal values for α_i and $\mathbf{u}_i$ when only local information of the function is available. Repeated application of eqn (13), each time with an optimal $\mathbf{u}_i$ and α_i, will bring E to a minimum. This minimum can be either local or global; there is no guarantee whatsoever about what kind of minimum is encountered. If the error landscape is complicated, that is, has many local minima, a better minimum may be sought by starting at another initial point, or perturbing the system away from the just found minimum in hope for a better one (Press et al., 1986; Aarts & Korst, 1989).

Line Minimisation. When the minimisation direction is available, the problem is to decide *how far* to go along this direction before a new direction is chosen. Usually, evaluations of the function and its derivatives are used to locate some minimum, global or local, and stop there. There are methods available (Bromberg & Chang, 1992) that locate the global minimum, but these methods require several tens to hundreds of function evaluations and are therefore too expensive. Also, it can be argued that perfect line minimisation may lead to getting stuck in local minima of the overall error landscape.

One often used method, presented by Press et al. (1986), is modeled after Brent: given there are three values $\mathbf{x}_a < \mathbf{x}_b < \mathbf{x}_c$ such that the function at $\mathbf{x}_b$ is the lowest (i.e., a minimum is bracketed by $\mathbf{x}_a$ and $\mathbf{x}_c$). The sign of the derivative at $\mathbf{x}_b$ indicates whether a minimum is located in $[\mathbf{x}_a, \mathbf{x}_b]$ or in $[\mathbf{x}_b, \mathbf{x}_c]$. This section is then linearly interpolated from its endpoints, and the procedure is repeated. This method is used in the simulations presented below.

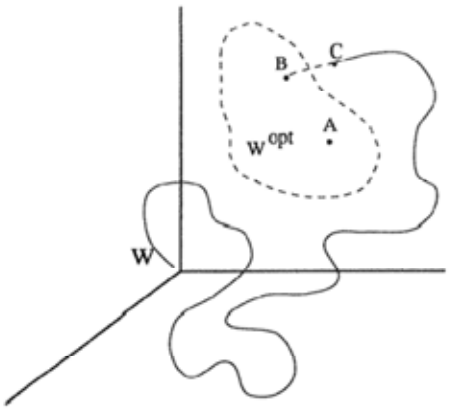

FIGURE 2. Learning of a feedforward neural network as a trajectory in weight space. Point A represents the best approximation $\mathbf{w}^{\text{opt}}$, point B a single realisation of the optimal solution for a given learning set, and point C is the actual solution. The subset of optimal solutions is given by the dashed line. [From Vyšniauskas et al. (1992); reprinted with permission.]

4.1. First-Order Methods

The learning error eqn (12) can be written as a Taylor expansion around $\mathbf{w}_0$,

$$E(\mathbf{w}) = E(\mathbf{w}_0) + \sum_{l,i,j} \left.\frac{\partial E}{\partial w_{lij}}\right|_{\mathbf{w}_0} w_{ij} + \tfrac{1}{2} \sum_{l,i,jl',i',j'} \left.\frac{\partial^2 E}{\partial w_{lij}\partial w_{l'i'j'}}\right|_{\mathbf{w}_0} \times w_{lij}w_{l'i'j'} + \cdots. \quad (14)$$

In first-order methods, all but the first term of this Taylor expansion are ignored. These methods, where the local gradient alone determines the minimisation direction $\mathbf{u}$, are known as *steepest descent* or *gradient descent;* in feedforward neural network training they are known as *error back propagation.*

Steepest descent works as follows. When the system is in a state $\mathbf{w}_i$, the gradient $\mathbf{g}_i = \partial E/\partial \mathbf{w}_i$ is computed and a minimisation step in the direction $\mathbf{u}_i = -\mathbf{g}_i$ is performed. In normal steepest descent minimisation, a one-dimensional minimisation in the direction of $\mathbf{u}_i$ is performed such that a point $\mathbf{w}_{i+1}$ is reached where $\mathbf{g}_{i+1}$ is perpendicular to $\mathbf{u}_i$. The learning rule then becomes

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \alpha_i \mathbf{u}_i, \tag{15}$$

where the search direction is chosen as

$$\mathbf{u}_{i+1} = -\mathbf{g}_{i+1}. \tag{16}$$

In standard back propagation, the line minimisation is usually replaced by a fixed step size α. A reason for this is cost: one function evaluation for a feedforward network means forward propagation of all the learning patterns. Therefore, performing a line minimisation, which usually requires at least three to five function evaluations, is costly and may not improve the algorithm (see Section 5).

The back propagation search direction is usually augmented with a *momentum term* (Rumelhart, Hinton, & Williams, 1986):

$$\mathbf{u}_{i+1} = -\mathbf{g}_{i+1} + \beta_i \mathbf{u}_i. \tag{17}$$

This extra term is generally interpreted as avoiding oscillations. It will be shown below that adding the momentum term is wise when the values α_i and β_i are well chosen.

The inefficiency of steepest descent is due to the fact that the minimisation directions and step sizes are poorly chosen: unless the first step is chosen such that it leads directly to the minimum, steepest descent will zig-zag with many small steps.

4.2. Second-Order Methods

When using not only the first but also the second term from the Taylor expansion eqn (14), the error eqn (12) can be written as

$$E(\mathbf{w}) \approx \tilde{E}(\mathbf{w}) = E(\mathbf{w}_0) - \mathbf{b}^T\mathbf{w} + \tfrac{1}{2}\mathbf{w}^T A\mathbf{w} \tag{18}$$

where $\mathbf{w}$ is expressed relative to $\mathbf{w}_0$ and

$$\mathbf{b} = -\nabla E|_{\mathbf{w}_0}, \quad \text{and} \quad [A]_{ij} = \left.\frac{\partial^2 E}{\partial w_i \partial w_j}\right|_{\mathbf{w}_0}.$$

A, the matrix of second derivatives, is called the *Hessian* of E at $\mathbf{w}_0$.

Minima are located where the gradient to eqn (18) is 0, that is,

$$\nabla \tilde{E} = A\mathbf{w} - \mathbf{b} = 0. \tag{19}$$

The optimal $\mathbf{w}^{\text{opt}}$ is then found to be

$$\mathbf{w}^{\text{opt}} = A^{-1}\mathbf{b}. \tag{20}$$

Thus, knowledge of the Hessian and gradient of E reduce the minimisation to matrix inversion, if $E - \tilde{E}$ is small. However, among other problems (Battiti, 1992), calculation of A is computationally prohibitive. We will therefore have to revert to approximating methods.

4.2.1. *Quasi-Newton Methods.* The aim of quasi-Newton methods, such as the BFGS (Broyden–Fletcher–Goldfarb–Shanno) and DFP (Davidon–Fletcher–Powell) methods, is to iteratively compute matrices H_i such that

$$\lim_{i\to\infty} H_i = A^{-1}. \tag{21}$$

The term *quasi-Newton* applies if

$$H_{i+1}(\mathbf{g}_{i+1} - \mathbf{g}_i) = \mathbf{w}_{i+1} - \mathbf{w}_i \tag{22}$$

is satisfied (this is trivially true for $H = A^{-1}$, because $\mathbf{g} = A\mathbf{w} - \mathbf{b}$). The resulting H_i can then be used to find

$$\mathbf{w}_{i+1} = H_i\mathbf{b} \tag{23}$$

until a minimum is reached. It is easily seen that the DFP update formula for H_i,

$$H_{i+1} = H_i + \frac{(\mathbf{w}_{i+1} - \mathbf{w}_i) \times (\mathbf{w}_{i+1} - \mathbf{w}_i)}{(\mathbf{w}_{i+1} - \mathbf{w}_i)^T(\mathbf{g}_{i+1} - \mathbf{g}_i)} - \frac{[H_i(\mathbf{g}_{i+1} - \mathbf{g}_i)] \times [H_i(\mathbf{g}_{i+1} - \mathbf{g}_i)]}{(\mathbf{g}_{i+1} - \mathbf{g}_i)^T H_i(\mathbf{g}_{i+1} - \mathbf{g}_i)} \tag{24}$$

satisfies eqn (22); it can be shown to converge to A^{-1} (Polak, 1971). A variant, with a slightly different update for H_{i+1}, is the BFGS update.

A disadvantage of these methods, which can be especially cumbersome for neural network training due to the number of elements ν in $\mathbf{w}$ is that storage of the H_i is quadratic in the number of weights of the network.

4.2.2. *Conjugate Gradient Methods.* An alternative second-order minimisation technique is *conjugate gradient optimisation* (Stoer & Bulirsch, 1980; Press et al., 1986; Polak, 1971; Powell, 1977; van der Smagt & Kröse, 1991). The direction of minimisation is always chosen such that the minimisation steps in all previous directions are not spoiled. That is to say, when the direction $\mathbf{u}_i$ is chosen and a line minimisation is performed in this direction, leading to a point $\mathbf{w}_{i+1}$, then the gradient $\mathbf{g}_{i+1}$ at $\mathbf{w}_{i+1}$ must be perpendicular to $\mathbf{g}_i$, $\mathbf{g}_{i-1}, \ldots, \mathbf{g}_0$. The initial minimisation direction may be chosen randomly, but is usually set to $-\mathbf{g}_0$.

When such directions $\mathbf{u}_i$ can be found and the approximation of E by its second-order Taylor expansion is exact, the minimum will be located in ν steps, where ν is the number of free parameters of the system.

Suppose the initial direction of minimisation, which is started at $\mathbf{w}_0$, is $\mathbf{u}_0$. A line minimisation in the di-

rection of $\mathbf{u}_0$ results in a gradient at $\mathbf{w}_1$ perpendicular to $\mathbf{u}_0$. In general,

$$\mathbf{u}_i^T \mathbf{g}_{i+1} = 0. \tag{25}$$

Because we do not want to spoil this minimisation step in subsequent minimisations, the gradients of subsequent points of minimisation must also be perpendicular to $\mathbf{u}_i$:

$$\mathbf{u}_i^T \mathbf{g}_{i+2} = 0. \tag{26}$$

Therefore, with eqn (25) and eqn (26),

$$\mathbf{u}_i^T (\mathbf{g}_{i+2} - \mathbf{g}_{i+1}) = 0. \tag{27}$$

Now, $\mathbf{g}_{i+2} - \mathbf{g}_{i+1}$ is the change in the gradient as we move from $\mathbf{w}_{i+1}$ to $\mathbf{w}_{i+2}$. With eqn (18), the gradient of $\tilde{E}$ at $\mathbf{w}_i$ can be found to be

$$\mathbf{g}_i = A\mathbf{w}_i - \mathbf{b}. \tag{28}$$

Therefore,

$$0 = \mathbf{u}_i^T (\mathbf{g}_{i+2} - \mathbf{g}_{i+1}) = \mathbf{u}_i^T A(\mathbf{w}_{i+2} - \mathbf{w}_{i+1}) = \mathbf{u}_i^T A \mathbf{u}_{i+1} \alpha_{i+1} \tag{29}$$

or

$$\mathbf{u}_i^T A \mathbf{u}_{i+1} = 0. \tag{30}$$

When eqn (30) holds for two vectors $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ these vectors are said to be *conjugate*.

After, through line minimisation along $\mathbf{u}_i$, a point $\mathbf{w}_{i+1}$ is reached, the next minimisation direction is constructed using

$$\mathbf{u}_{i+1} = -\mathbf{g}_{i+1} + \beta_i \mathbf{u}_i. \tag{31}$$

Conjugacy of $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ is obtained when

$$\beta_i = \frac{\mathbf{g}_{i+1}^T \mathbf{g}_{i+1}}{\mathbf{g}_i^T \mathbf{g}_i}. \tag{32}$$

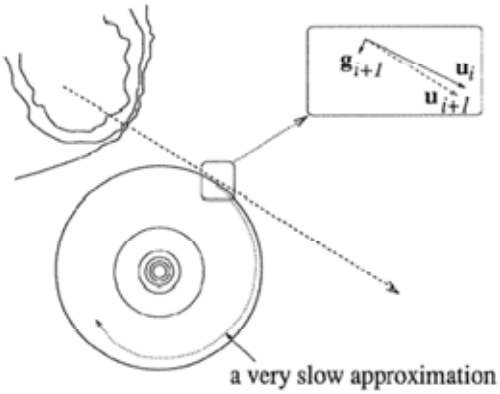

FIGURE 3. Slow decrease with conjugate gradient in nonquadratic systems. The hills on the left are very steep, resulting in a large search vector $\mathbf{u}_i$. When the quadratic portion is entered the new search direction is constructed from the previous direction and the gradient, resulting in a spiraling minimisation (van Summeren, 1990).

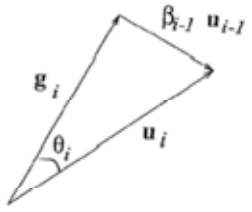

FIGURE 4. Definition of θ_i.

It is easy to show, if we take

$$\alpha_i = \frac{\mathbf{g}_i^T \mathbf{g}_i}{\mathbf{u}_i^T A \mathbf{u}_i} \tag{33}$$

and

$$\mathbf{g}_{i+1} = \mathbf{g}_i - \alpha_i A \mathbf{u}_i, \tag{34}$$

that $\mathbf{u}_i^T A \mathbf{u}_{j \neq i} = 0$ ($\mathbf{u}_i$ and $\mathbf{u}_j$ conjugate) and that $\mathbf{g}_i^T \mathbf{g}_{j \neq i} = 0$ ($\mathbf{g}_i$ and $\mathbf{g}_j$ orthogonal). This method is known as *Fletcher–Reeves conjugate gradient*.

It is immediately concluded that eqns (31)–(33) implement the error back propagation learning rules eqn (15) and eqn (17) with learning parameter α and momentum β thus chosen that subsequent search directions are conjugate. Therefore, conjugate gradient is a special case of error back propagation with momentum term, such that second-order information is used.

A closer look at eqn (33) shows that α is large when $\det(A)$ is small, that is, there is a small second-order component in the Taylor approximation of E. In this case, where the function is almost lincar, thc minimisation direction follows the gradient. When the quadratic component is more prominent, the minimisation takes a more conservative approach.

Equations (33) and (34) both require the Hessian A. This embarrassment, because A is not known, can be solved by line minimisation; the α and $\mathbf{g}$ obtained are the same. Thus, the need of the matrix A or its inverse is eliminated.

4.2.3. *Improvements of Conjugate Gradient.* Although only ν iterations are needed for a quadratic system with

TABLE 1
Percentage of Runs That Lead to a Global Minimum

	XOR	sin(x)cos(2x)	tan(x)
BP	91.3	15*	0.0
SD	38.0	90	0.0
FR	81.5	49.5	4.5
DFP	34.1	35.7	40.0
CG	82.1	100.0	85.4

A global minimum was considered reached when the summed squared error was less than 0.025 per pattern.

* Indicates that adaptive learning rates were used in this case.

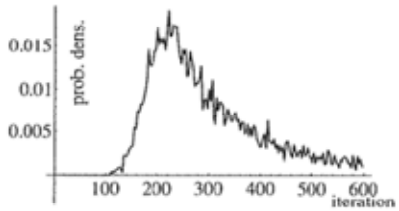

FIGURE 5. Probability density of the number of function evaluations required to reach a global minimum (summed squared error less than 0.025 per pattern) for XOR classification with standard error back propagation.

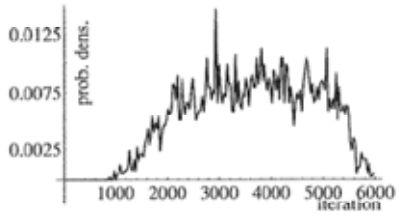

FIGURE 6. Probability density of the number of function evaluations required to reach a global minimum (summed squared error less than 0.025 per pattern) for XOR classification with error back propagation and line minimisation (steepest descent).

ν degrees of freedom, due to the fact that we are not minimising quadratic systems, as well as a result of round off errors, the ν directions have to be followed several times, with a restart of $\mathbf{u}_{\nu+1} = -\mathbf{g}_{\nu+1}$. Restarting the minimisation method requires some thought, and improvements have been suggested by Polak and Ribière (1971) and Powell (1977). The resulting cost of the minimisation procedure is $O(\nu)$, which is significantly better than the linear convergence[3] of steepest descent.

The Value of β_i. Equation (31) can give erroneous results when the system is in a region where it is not nearly quadratic. Consider the case in Figure 3. When the search direction $\mathbf{u}_i$ is very large, and a quadratic portion near the minimum is entered where the gradient is small, a spiraling minimisation will ensue. The slow spiraling can be detected as follows. At each new minimisation step, the angle θ_i between $\mathbf{g}_i$ and $\mathbf{u}_i$ (see Figure 4) equals

$$\|\mathbf{u}_i\| = \frac{\|\mathbf{g}_i\|}{\cos\theta_i}. \tag{35}$$

Replacing i by $i+1$ in Figure 4 results in

$$\beta_i \|\mathbf{u}_i\| = \tan\theta_{i+1} \|\mathbf{g}_{i+1}\|. \tag{36}$$

$\|\mathbf{u}_i\|$ can be eliminated from eqns (35) and (36):

$$\tan\theta_{i+1} = \frac{1}{\cos\theta_i} \frac{\|\mathbf{g}_{i+1}\|}{\|\mathbf{g}_i\|} > \tan\theta_i \frac{\|\mathbf{g}_{i+1}\|}{\|\mathbf{g}_i\|}. \tag{37}$$

Now, if θ_i approaches $\pi/2$, the iteration may take a small step such that $(\mathbf{g}_{i+1} - \mathbf{g}_i)$ is small and the ratio $\|\mathbf{g}_{i+1}\|/\|\mathbf{g}_i\|$ approaches unity. Then, according to eqn (37), θ_{i+1} is also near $\pi/2$ and the effect depicted in Figure 3 is observed. If, however, the expression for β_i in eqn (32) is replaced by

$$\beta_i = \frac{\mathbf{g}_{i+1}^T(\mathbf{g}_{i+1} - \mathbf{g}_i)}{\mathbf{g}_i^T \mathbf{g}_i}, \tag{38}$$

then

$$\beta_i \le \frac{\|\mathbf{g}_{i+1}\| \cdot \|\mathbf{g}_{i+1} - \mathbf{g}_i\|}{\|\mathbf{g}_i\|^2} \tag{39}$$

or

$$\tan\theta_{i+1} \le \frac{1}{\cos\theta_i} \frac{\|\mathbf{g}_{i+1} - \mathbf{g}_i\|}{\|\mathbf{g}_i\|} \tag{40}$$

such that $\theta_{i+1} \le \theta_i$ and $\mathbf{u}_{i+1}$ is turned towards the steepest descent direction. Notice that this expression for β_i retains the conjugacy of $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$.

Conjugate gradient minimisation with eqn (38) is known as Polak–Ribière conjugate gradient.

Powell's Restart Procedures. A second improvement involves the restart. It appears that restarting with $\mathbf{u}_{\nu+1} = -\mathbf{g}_{\nu+1}$ is inefficient. Instead, a restarting method that does not abandon the second derivative information is needed. A proposed method is obtained by setting $\mathbf{u}_i$ using eqn (31) when a restart is made, and by extending the definition of $\mathbf{u}_i$ on normal iterations (Beale, 1972).

Let $\mathbf{u}_k$ be an arbitrary downhill restarting direction. Supposing E is quadratic, we are looking for a direction $\mathbf{u}_{i+1}$ that is a linear combination of $\mathbf{u}_k$ and the gradients $\mathbf{g}_k, \mathbf{g}_{k+1}, \ldots, \mathbf{g}_{i+1}$ such that $\mathbf{u}_k, \mathbf{u}_{k+1}, \ldots$ are mutually conjugate. An expression that suffices these conditions is

$$\mathbf{u}_{i+1} = -\mathbf{g}_{i+1} + \beta_i \mathbf{u}_i + \beta_i' \mathbf{u}_k. \tag{41}$$

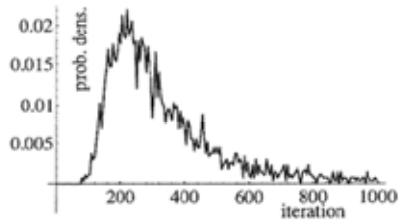

FIGURE 7. Probability density of the number of function evaluations required to reach a global minimum (summed squared error less than 0.025 per pattern) for XOR classification with quasi-Newton DFP.

[3] A method is said to converge linearly if the error $E_{i+1} = cE_i$ with $c < 1$, where E is the error still left over. Methods that converge with a higher power, that is, $E_{i+1} = c(E_i)^m$ with $m > 1$, are called *super-linear.*

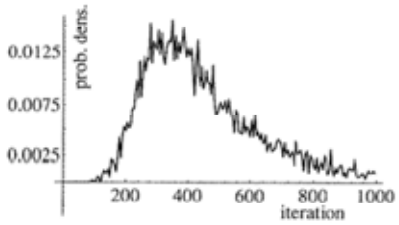

FIGURE 8. Probability density of the number of function evaluations required to reach a global minimum (summed squared error less than 0.025 per pattern) for XOR classification with Fletcher–Reeves conjugate gradient.

Again, β_i is calculated to make $\mathbf{u}_{i+1}$ conjugate to $\mathbf{u}_i$, and the extra term provides conjugacy to $\mathbf{u}_k$:

$$\beta_i = \frac{\mathbf{g}_{i+1}^T(\mathbf{g}_{i+1} - \mathbf{g}_i)}{\mathbf{u}_i^T(\mathbf{g}_{i+1} - \mathbf{g}_i)}, \quad \beta_i' = \frac{\mathbf{g}_{i+1}^T(\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{u}_{i+1}^T(\mathbf{g}_{k+1} - \mathbf{g}_k)}. \quad (42)$$

To prevent that the resulting direction leads uphill instead of downhill, we require that $\mathbf{u}_i^T\mathbf{g}_i > 0$ for $i > k$. Furthermore, the orthogonality between $\mathbf{g}_{i-1}$ and $\mathbf{g}_i$ must be guaranteed after restart to prevent that the approximations go to a nonzero limit:

$$\mathbf{g}_{i-1}^T\mathbf{g}_i \leq 0.2\|\mathbf{g}_i\|^2. \quad (43)$$

Thirdly, the new search direction must go *sufficiently* downhill:

$$-1.2\|\mathbf{g}_i\|^2 \leq \mathbf{u}_i^T\mathbf{g}_i \leq -0.8\|\mathbf{g}_i\|^2. \quad (44)$$

If eqns (43) or (44) are not satisfied, we restart with $i = k - 1$. The mentioned constants are suggested by Powell.

5. RESULTS

Where appropriate, the following methods are compared:

BP: standard error back propagation with fixed learning rates;

SD: error back propagation with line minimisation instead of learning rates (i.e., steepest descent minimisation);

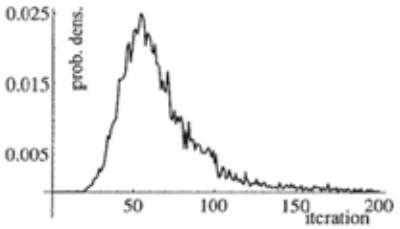

FIGURE 9. Probability density of the number of function evaluations required to reach a global minimum (summed squared error less than 0.025 per pattern) for XOR classification with Polak–Ribière conjugate gradient with Powell restarts.

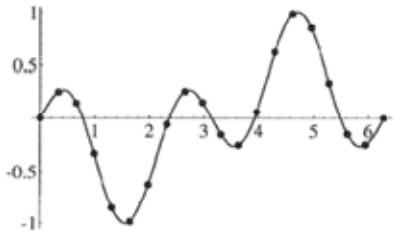

FIGURE 10. Learning samples from the function sin(x)cos(2x).

DFP: Davidon–Fletcher–Powell quasi-Newton minimisation;

FR: Fletcher–Reeves conjugate gradient minimisation;

CG: conjugate gradient minimisation with Powell restarts.

The number of necessary error function evaluations is counted as the number of iterations. With error back propagation, this equals the number of search directions traversed. For the other methods, due to line minimisation every search direction is traversed multiple times, and in each instance the function value as well as the derivative is computed.

The methods are compared on the following three problems:

1. classification of the four XOR patterns, to reach an average squared error of less than 0.025 per pattern;
2. approximation of the function $\sin(x)\cos(2x)$ for $0 \leq x \leq 2\pi$, from which 20 samples are uniformly chosen. An average squared error less than 0.025 per pattern must be reached;
3. approximation of the function $\tan(x)$ (with one discontinuity) for $0 \leq x \leq \pi$, from which 20 samples are uniformly chosen. An average squared error less than 0.025 per pattern must be reached.

In all instances, 10,000 trials were run.

5.1. Classification the XOR Problem

It has been observed (Fahlman, 1988) that the exclusive or classification problem is not a representative problem because it does not encourage but rather penalises when

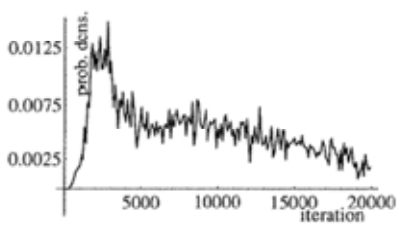

FIGURE 11. Probability density of the number of function evaluations required to reach a global minimum (summed squared error less than 0.025 per pattern) for sin(x)cos(2x) approximation with Fletcher–Reeves conjugate gradient.

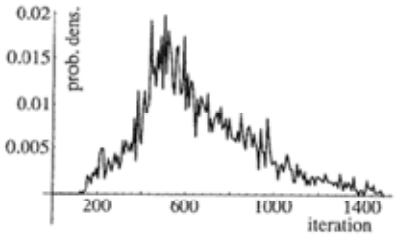

FIGURE 12. Probability density of the number of function evaluations required to reach a global minimum (summed squared error less than 0.025 per pattern) for sin(x)cos(2x) approximation with quasi-Newton DFP.

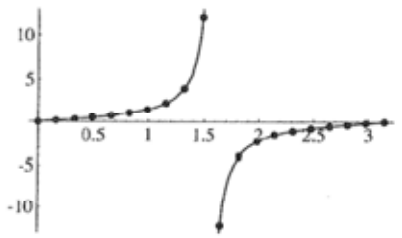

FIGURE 14. Learning samples from the function tan(x).

the network tries to generalise the learning patterns. However, it is a difficult classification problem, and therefore we use it as a benchmark for the five minimisation methods.

For all learning methods, 10,000 trials were run with a network with two hidden units in one layer. For those runs that reached a global minimum, where a global minimum is considered reached when the summed squared error is less than 0.025 per pattern, the number of iterations needed is counted. Table 1 shows how often a global minimum is reached.

First, standard error back propagation is tested with a learning rate of 0.1 and a momentum of 0.9. In 91.3% of the trials a global minimum is reached. Figure 5 shows the number of iterations needed to reach a summed squared error of less than 0.025 per pattern; the average is located at 332. When, each step, a line minimisation is performed, the system is much more sensitive to local minima: only in 38.0% of all trials was a global minimum reached. The number of steps needed is shown in Figure 6. Notice, however, that each minimisation now takes three to five line minimisations; the average number of minimisation steps is therefore the average number of function evaluations (3661.7) divided by the average number of line minimisation steps (three to five).

The quasi-Newton method reaches a global minimum only in 34.1% of the cases, and the average number of learning iterations is 2141.1 (see Figure 7). Fletcher–Reeves conjugate gradient reaches the minimum in 81.5% of the cases, but needs, on the average, 523.0 iterations (see Figure 8). Finally, conjugate gradient with Powell restarts is 82.1% successful, and only needs 79.2 iterations (Figure 9).

It is clear that, to locate the "region" of the global minimum, standard error back propagation is very successful when, initially, line search tends to move towards local minima, and precision in locating the line minimum has no positive influence.

5.2. Continuous Function Approximation

Secondly, the function $\sin(x)\cos(2x)$ was trained with 20 learning samples over the period $0 \ldots 2\pi$ (see Figure 10). The used network had 10 hidden units in one layer (Vyšniauskas et al., 1992).

Figures 11–13 show the number of function evaluations needed to reach a summed squared error less than 0.025 per pattern. Although DFP is slightly better than CG, the latter always reaches a global minimum, whereas the former only reaches a minimum in 35% of the cases.

Steepest descent is not very sensitive to local minima here, but needs, on the average, $4 \cdot 10^6$ iterations to reach the minimum.

Error back propagation was, when fixed stepsizes were used, never able to find the minimum; when the system was approaching a global minimum, the stepsize was not small enough to continue decreasing the error,

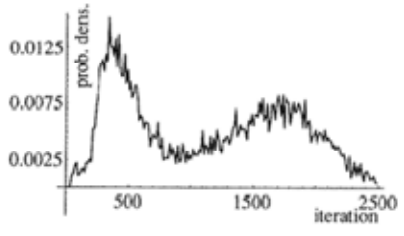

FIGURE 13. Probability density of the number of function evaluations required to reach a global minimum (summed squared error less than 0.025 per pattern) for sin(x)cos(2x) approximation with Polak–Ribière conjugate gradient with Powell restarts.

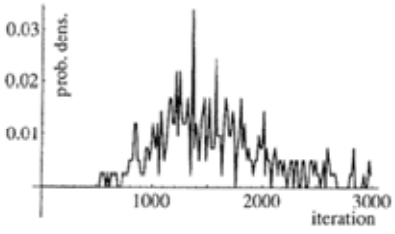

FIGURE 15. Probability density of the number of function evaluations required to reach a global minimum (summed squared error less than 0.025 per pattern) for tan(x) approximation with Fletcher–Reeves conjugate gradient.

such that overshoot resulted. To overcome this problem, adaptive learning rates (Silva & Almeida, 1990) were used. Still the algorithm gave poor results, often reaching local minima and needing over two million function evaluations.

5.3. Discontinuous Function Approximation

Finally, the function $\tan(x)$ over $0 \ldots \pi$, with one discontinuity, was tested with 20 learning samples equally distributed over the inputs space (see Figure 14). A network with five hidden units in one hidden layer was used (Vyšniauskas et al., 1992). Steepest descent and error back propagation never reached a global minimum. The other methods (see Figures 15–17) could solve the problem.

6. DISCUSSION

It has been shown that, although standard error back propagation is less sensitive to get stuck in local minima, second-order minimisation methods are far superior with respect to learning time, especially in accurately approximating smooth functions. Conjugate gradient algorithms, which can be seen as error back propagation with momentum, were shown to be a better choice for feedforward network training. In particular, Polak–Ribière conjugate gradient optimisation with Powell restarts shows promising results for training feedforward networks.

The problem of local minima is serious, and further investigation to the nature of the problem is required. Some authors suggest hybrid training algorithms (Gorse & Shepherd, 1992): switch to conjugate gradient optimisation near minima, and use error back propagation with fixed stepsize otherwise. This method, which is also suggested by Møller (1990), is a technique from numerical analysis known as Levenberg–Marquardt optimisation (Press et al., 1986).

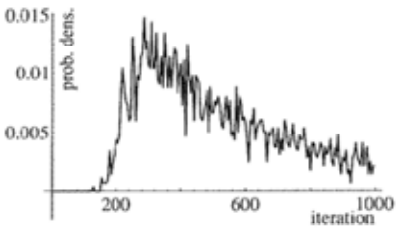

FIGURE 16. Probability density of the number of function evaluations required to reach a global minimum (summed squared error less than 0.025 per pattern) for tan(x) approximation with quasi-Newton DFP.

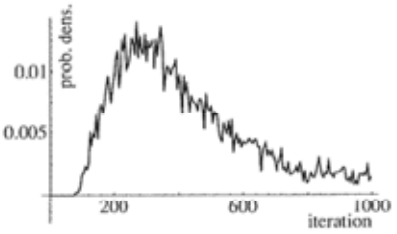

FIGURE 17. Probability density of the number of function evaluations required to reach a global minimum (summed squared error less than 0.025 per pattern) for tan(x) approximation with Polak–Ribière conjugate gradient with Powell restarts.

REFERENCES

Aarts, E., & Korst, J. (1989). *Simulated annealing and Boltzmann machines.* New York: John Wiley & Sons.

Barnard, E. (1992). Optimization for training neural nets. *IEEE Transactions on Neural Networks,* **3**(2), 232–240.

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

Beale, E. M. L. (1972). A derivation of conjugate gradients. In F. A. Lootsma (Ed.), *Numerical methods for nonlinear optimization.* London: Academic Press.

Bromberg, M., & Chang, T.-S. (1992). One dimensional global optimization using linear lower bounds. In C. A. Floudas & P. M. Pardalos (Eds.), *Recent advances in global optimization* (pp. 200–220). Princeton, NJ: Princeton University Press.

Chen, A. M., & Hecht-Nielsen, R. (1989). On the geometry of feedforward neural network weight spaces. In *Proceedings of the Second IEEE International Conference on Neural Networks,* 1–4.

Fahlman, S. E. (1988). An empirical study of learning speed in backpropagation networks (Tech. Rep. CMU-CS-88-0-162). Pittsburgh, PA: Carnegie Mellon University.

Forsythe, G. E. (1957). Generation and use of orthogonal polynomials for data-fitting with a digital computer. *Journal of the Society of Industrial and Applied Mathematics,* **5**(2), 74–88.

Gorse, D., & Shepherd, A. (1992). Adding stochastic search to conjugate gradient algorithms. In *Proceedings of the 3rd International Conference on Parallel Applications in Statistics and Economics.* Prague: Tiskárenské Závody.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks,* **2**(5), 359–366.

Kadirkamanathan, V., & Fallside, F. (1990). F-projections: A nonlinear recursive estimation algorithm for neural networks (Tech. Rep. CUED/F-INFENG/TR.53). Cambridge, UK: Cambridge University Engineering Department.

Kinsella, J. A. (1992). Comparison and evaluation of variants of the conjugate gradient method for efficient learning in feed-forward neural networks with backward error propagation. *Network,* **3,** 27–35.

Miller III, W. T. (1989). Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Transactions on Systems, Man, and Cybernetics,* **19**(4), 825–831.

Møller, M. F. (1990). A scaled conjugate gradient algorithm for fast supervised learning (Tech. Rep. PB-339). Aarhus, Denmark: University of Aarhus, Computer Science Department.

Polak, E. (1971). *Computational methods in optimization.* New York; Academic Press.

Powell, M. J. D. (1977). Restart procedures for the conjugate gradient method. *Mathematical Programming,* **12,** 241–254.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1986). *Numerical recipes: The art of scientific computing.* Cambridge: Cambridge University Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature,* **323,** 533–536.

Silva, F. M., & Almeida, L. B. (1990). Speeding up backpropagation. In R. Eckmiller (Ed.), *Advanced neural computers* (pp. 151–160). Amsterdam: North–Holland.

Stoer, J., & Bulirsch, R. (1980). *Introduction to numerical analysis.* New York: Springer–Verlag.

van der Smagt, P. P., Jansen, A., & Groen, F. C. A. (1992). Interpolative robot control with the nested network approach. In *Proceedings of the 1992 IEEE International Symposium on Intelligent Control* (pp. 475–480). Glasgow, Scotland: IEEE Control Systems Society.

van der Smagt, P. P., & Kröse, B. J. A. (1991). A real-time neural robot controller. In *Proceedings of the 1991 International Conference on Artificial Neural Networks* (pp. 351–356). Espoo, Finland: Elsevier Science Publishers.

Vyšniauskas, V., Groen, F. C. A., & Kröse, B. J. A. (1992). *Function approximation with a feedforward network: The optimal number of learning samples and hidden units.* Department of Computer Systems, University of Amsterdam, Amsterdam.

APPENDIX

Orthogonal vs. Nonorthogonal Basis Functions

When a set of one-dimensional data points (x_p, y_p) is fitted with polynomials, the most obvious choice of basis functions would be $\phi_i(x) = x^i$. The set of P data points then can be fitted by minimising the error function, also known as chi-square merit function

$$E = \sum_{p=1}^{P} \left[y_p - \sum_{j=0}^{n} a_j x_p^j \right]^2 \tag{45}$$

where n is the desired degree of the approximation. Minimum values for eqn (45) are reached where its derivatives vanish, that is,

$$0 = \sum_{p=1}^{P} \left[y_p - \sum_{j=0}^{n} a_j x_p^j \right] x_p^k, \quad k = 0, \ldots, n. \tag{46}$$

Solving eqn (46) involves inversion of the matrix A with

$$A_{ij} = \sum_{p=1}^{P} x_p^i x_p^j \approx P \int_0^1 x^i x^j = \frac{P}{i + j + 1}. \tag{47}$$

The matrix A, written like this, is the principal minor of order $n + 1$ of the infinite Hilbert matrix

$$H = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

that is extremely difficult to invert for large n, because roundoff errors will have a tremendous impact.

When orthogonal basis functions are used, however, all the elements of A except those on the diagonal disappear, and inversion reduces to a simple scalar division.

NOMENCLATURE

$\mathcal{F} : \Re^m \to \Re^n$	the function that will be approximated
$S^P = \{s^1, s^2, \ldots, s^P\}$	the set of learning samples with $s^p = [\mathbf{x}^p, \mathcal{F}(\mathbf{x}^p)]$
$\mathbf{w}$	the weights in the network. There is a total of ν parameters. The optimal weight vector is denoted by $\mathbf{w}^{\text{opt}}$
$\phi_i : \Re \to \Re,\ 1 \le i \le \kappa$	the κ basis functions
$\mathcal{N}_{\mathbf{w}}^{\mathbf{w}} : \Re^m \to \Re^n$	the network that is used for approximating $\mathcal{F}$