NEURAL NETWORK SUPERVISED TRAINING BASED ON A DIMENSION REDUCING METHOD

G.D. Magoulas, M.N. Vrahatis*,
T.N. Grapsa* and G.S. Androulakis*
Department of Electrical and Computer Engineering, University of Patras, GR-261.10, Patras, Greece. Email: magoulas@ee-gw.ee.upatras.gr
** Department of Mathematics, University of Patras, GR-261.10 Patras, Greece. Email: vrahatis—grapsa—gsa@math.upatras.gr*

In this contribution a new method for supervised training is presented. This method is based on a recently proposed root finding procedure for the numerical solution of systems of non–linear algebraic and/or transcendental equations in $\mathbb{R}^n$. This new method reduces the dimensionality of the problem in such a way that it can lead to an iterative approximate formula for the computation of $n-1$ connection weights. The remaining connection weight is evaluated separately using the final approximations of the others. This reduced iterative formula generates a sequence of points in $\mathbb{R}^{n-1}$ which converges quadratically to the proper $n-1$ connection weights. Moreover, it requires neither a good initial guess for one connection weight nor accurate error function evaluations. The new method is applied on some test cases in order to evaluate its performance.
Subject classification: AMS(MOS) 65K10, 49D10, 68T05, 68G05.

Keywords: Numerical optimization methods, feed forward neural networks, supervised training, back–propagation of error, dimension–reducing method.

1 Introduction

Consider a feed forward neural network (FNN) with l layers, $l \in [1, L]$. The error is defined as $e_k(t) = d_k(t) - y_k^L(t)$, for $k = 1, 2, ..., K$, where $d_k(t)$ is the desired response at the kth neuron of the output layer at the input pattern t, $y_k^L(t)$ is the output at the kth neuron of the output layer L. If there is a fixed, finite set of input–output cases, the square error over the training set which contains T representative cases is:

$$E = \sum_{t=1}^{T} E(t) = \sum_{t=1}^{T} \sum_{k=1}^{K} e_k^2(t). \tag{1}$$

The most common supervised training algorithm for FNNs with sigmoidal non–linear neurons is the Back–Propagation (BP), [4]. The BP minimizes the error function E using the Steepest Descent (SD) with fixed step size and computes the gradient using the chain rule on the layers of the network. BP converges too slow and often yields suboptimal solutions. The quasi–Newton method (BFGS) [2], converges much faster than the BP but the storage and computational requirements of the Hessian for very large FNNs make its use impractical for most current machines.
In this paper, we derive and apply a new training method for FNNs named Dimension Reducing Training Method (DRTM). DRTM is based on the methods studied in [3] and it incorporates the advantages of Newton and SOR algorithms (see [4]).

2 Description of the DRTM

Throughout this paper $\mathbb{R}^n$ is the n–dimensional real space of column weight vectors w with components $w_1, w_2, \ldots, w_n$; $(y; z)$ represents the column vector with components $y_1, y_2, \ldots, y_m, z_1, z_2, \ldots, z_k$; $\partial_i E(w)$ denotes the partial derivative of $E(w)$ with respect to the ith variable w_i; $g(w) = (g_1(w), \ldots, g_n(w))$ defines the gradient $\nabla E(w)$ of the objective function E at w while $H = [H_{ij}]$ defines the Hessian

S. W. Ellacott et al. (eds.), *Mathematics Of Neural Networks*

$\nabla^2 E(w)$ of E at w; $\bar{A}$ denotes the closure of the set A and $E(w_1, \ldots, w_{i-1}, \cdot, w_{i+1}, \ldots, w_n)$ defines the error function obtained by holding $w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n$ fixed.

The problem of training is treated as an optimization problem in the FNN's weight space (i.e., n–dimensional Euclidean space). In other words, we want to find the proper weights that satisfy the following system of equations :

$$g_i(w) = 0, \quad i = 1, \ldots, n. \tag{2}$$

In order to solve this system iteratively we want a sequence of weight vectors $\{w^p\}, p = 0, 1, \ldots$ which converges to the point $w^* = (w_1^*, \ldots, w_n^*) \in \mathcal{D} \subset \mathbb{R}^n$ of the function E. First, we consider the sets $\mathcal{B}_i$, to be those connected components of $g_i^{-1}(0)$ containing w^* on which $\partial_n g_i \neq 0$, for $i = 1, \ldots, n$ respectively. Next, applying the Implicit Function Theorem (see [4, 3]) for each one of the components g_i we can find open neighborhoods $\mathcal{A}_1^* \subset \mathbb{R}^{n-1}$ and $\mathcal{A}_{2,i}^* \subset \mathbb{R}$ of the points $y^* = (w_1^*, \ldots, w_{n-1}^*)$ and w_n^* respectively, such that for any $y = (w_1, \ldots, w_{n-1}) \in \bar{\mathcal{A}}_1^*$ there exist unique mappings φ_i defined and continuous in $\mathcal{A}_1^*$ such that : $w_n = \varphi_i(y) \in \bar{\mathcal{A}}_{2,i}^*$, and $g_i\big(y; \varphi_i(y)\big) = 0, \quad i = 1, \ldots, n$. Moreover, the partial derivatives $\partial_j \varphi_i, j = 1, \ldots, n-1$ exist in $\mathcal{A}_1^*$ for each φ_i, they are continuous in $\bar{\mathcal{A}}_1^*$ and they are given by :

$$\partial_j \varphi_i(y) = -\frac{\partial_j g_i\big(y; \varphi_i(y)\big)}{\partial_n g_i\big(y; \varphi_i(y)\big)}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n-1. \tag{3}$$

Working exactly as in [3], we utilize Taylor's formula to expand $\varphi_i(y)$, about y^p. By straightforward calculations, utilizing approximate values for $g_i(\cdot)$ and $\partial_j g_i(\cdot) \equiv \partial_{ij}^2 E$ (see [5], where error estimates for these approximations can also be found) we obtain the following iterative scheme for the computation of the $n-1$ components of w^* :

$$y^{p+1} = y^p + A_p^{-1} V_p, \quad p = 0, 1, \ldots, \tag{4}$$

where $y^p = [w_i^p], V_p = [v_i] = [w_n^{p,i} - w_n^{p,n}]$ and the elements of the matrix A_p are :

$$[a_{ij}] = \left[\frac{g_i(y^p + he_j; w_n^{p,i}) - g_i(y^p; w_n^{p,i})}{g_i(y^p; w_n^{p,i} + he_n) - g_i(y^p; w_n^{p,i})} - \frac{g_n(y^p + he_j; w_n^{p,n}) - g_n(y^p; w_n^{p,n})}{g_n(y^p; w_n^{p,n} + he_n) - g_n(y^p; w_n^{p,n})} \right], \tag{5}$$

with $w_n^{p,i} = \varphi_i(y^p)$, h a small quantity and e_j the j–th unit vector. After a desired number of iterations of (4), say $p = m$, the nth component of w^* can be approximated by means of the following relation :

$$w_n^{m+1} = w_n^{m,n} - \sum_{j=1}^{n-1} \left\{ (w_j^{m+1} - w_j^m) \cdot \frac{g_n(y^m + he_j; w_n^{m,n}) - g_n(y^m; w_n^{m,n})}{g_n(y^m; w_n^{m,n} + he_n) - g_n(y^m; w_n^{m,n})} \right\}. \tag{6}$$

Note that the iterative formula (4) uses the matrices A_p and V_p. The matrix A_p constitutes the reduced–Hessian of our network and its components incorporate components of the Hessian but are evaluated at different points. The matrix V_p uses only the points $w_n^{p,i}$ $(i = 1, \ldots, n-1)$ and $w_n^{p,n}$ instead of the gradient values employed in Newton's method. A proof for the convergence of (4) and (6) can be found in [6].

Relative procedures for obtaining w^* can be constructed by replacing w_n with any one of the components $w_1, \ldots, w_{n-1}$, for example w_{int}. The above described method

	FR		*PR*		*BFGS*		*DRTM*		
w^0	*IT*	*FE*	*IT*	*FE*	*IT*	*FE*	*IT*	*FE*	*ASG*
$(0.3, 0.4)$	F	F	F	F	F	F	5	20	100
$(-1, -2)$	F	F	F	F	14	274	7	28	140
$(-1, 10)$	F	F	F	F	14	285	7	28	140
$(0.2, 0.2)$	F	F	F	F	F	F	5	20	100
$(2, 1)$	F	F	F	F	13	298	5	20	100
$(0.3, 0.3)$	F	F	F	F	F	F	5	20	100
$(-1.2, 1.2)$	F	F	F	F	F	F	7	28	140

Table 1 Comparative results for Example 1.

does not require the expressions φ_i but only the values $w_n^{p,i}$ which are given by the solution of the one–dimensional equations $g_i(w_1^p, \ldots, w_{n-1}^p, \cdot) = 0$. So, by holding $y^p = (w_1^p, \ldots, w_{n-1}^p)$ fixed, we can solve the equations : $g_i(y^p; r_i^p) = 0, \ \ i = 1, \ldots, n$, for an approximate solution r_i^p in the interval (a, b) with an accuracy D. In order to solve the one–dimensional equations, we employ a modified bisection method described in [3, 12] and given by the following formula :

$$w^{p+1} = w^p + \mathrm{sgn}\psi(w^p)\, q \,/\, 2^{p+1}, \qquad p = 0, 1, \ldots, \tag{7}$$

with $w^0 = a$, $q = \mathrm{sgn}\psi(a)\,(b-a)$ and where sgn defines the well known sign function. This method computes with certainty a root when $\mathrm{sgn}\psi(w^0)\,\mathrm{sgn}\psi(w^p) = -1$ (see [12]). It is evident from (7) that the only computable information required by this method is the algebraic signs of the function ψ.
A high–level description of the new algorithm can be found in [8].

3 Simulation Results

Here we present and compare the behavior of the DRTM with other popular methods on some artificially created but characteristic situations. For example, it is common in FNN training to take minimization steps that increase some weights by large amounts pushing the output of the neuron into saturation. Moreover, in various small and large scale neural network applications the error surface has flat and steep regions. It is well known that the BP is highly inefficient in locating minima in such surfaces. In the following examples, the gradient is evaluated using finite differences for the DRTM and analytically for all the other methods.

Example 1 *The objective function's surface has flat and steep regions*

$$E(w) = \sum_{i=1}^{10} g_i^2, \qquad g_i(w_1, w_2) = 2 + 2i - \left(e^{iw_1} + e^{iw_2}\right). \tag{8}$$

System (8), which is a well–known test case, *(Jennrich and Sampson Function)* (see [9]), has a global minimum at $w_1 = w_2 = 0.2578\ldots$. In Table 1 we present results obtained by applying the nonlinear conjugate gradient methods Fletcher–Reeves (FR) and Polak–Ribiere (PR) and the quasi–Newton Broyden–Fletcher–Goldfarb–Shanno (BFGS) method with the corresponding numerical results of DRTM. In this Table *IT* indicates the total number of iterations required to obtain w^* (iterations

BP			*DRTM*			
MN	*STD*	*SUC*	*MN*	*STD*	*SUC*	*MAS*
100.4	77.3	38.5	2.3	1.2	72.5	67.8

Table 2 Comparison of Back-propagation with DRTM for Example 2.

limit= 500); *FE* the total number of function evaluations (and derivatives) and *ASG* the total number of algebraic signs of the components of the gradient that are required for applying the iterative scheme (7). Because of the difficulty of the problem FR and PR failed to converge in all the cases (marked with an *F* in the table). The results are mixed with the BFGS method. Especially, when we are close to the minimum BFGS leaves the appropriate region moving to wrong direction in order to minimize the objective function.

Example 2 *The objective function's surface is oval shaped and bent.*

We can artificially create such a surface by training a single neuron with sigmoid non-linearity using the patterns $\{-6,1\}$, $\{-6.1,1\}$, $\{-4.1,1\}$, $\{-4,1\}$, $\{4,1\}$, $\{4.1,1\}$, $\{6,1\}$, $\{6.1,1\}$ for input and $\{0\}$, $\{0\}$, $\{0.97\}$, $\{0.99\}$, $\{0.01\}$, $\{0.03\}$, $\{1\}$, $\{1\}$ for output. The weights w_1, w_2 take values in the interval $[-3,3] \times [-7.5,7.5]$. The global minimum is located at the center of the surface and there are two valleys that lead to local minima. The step size for the BP was 0.05. The initial weights were formed by spanning the interval $[-3,3]$ in steps of 0.05 and the interval $[-7.5,7.5]$ in steps of 0.125.

The behavior of the methods is exhibited in Table 2, where *MN* indicates the mean number of iterations for simulations that reached the global minimum; *STD* the standard deviation of iterations; *SUC* the percentage of success in locating the global minimum and *MAS* the mean number of algebraic signs that are required for applying the iterative scheme (7). Note that for DRTM, since finite differences are used, two error function evaluations are required in each iteration. BP succeeds to locate the global minimum when initial weights take values in the intervals $w_1 \in [-0.8,1.5]$ and $w_2 \in [-2.5,2.5]$. On the other hand, DRTM is less affected by the initial weights. In this case we exploit the fact that we are able to isolate the weight vector component most responsible for unstable behavior by reducing the dimension of the problem. Therefore, DRTM is very fast and possesses high percentage of success.

4 Conclusion and Further Improvements

This paper describes a new training method for FNNs. Although the proposed method uses reduction to simpler one-dimensional equations, it converges quadratically to $n-1$ components of an optimal weight vector, while the remaining weight is evaluated separately using the final approximations of the others. Thus, it does not require a good initial estimate for one component of an optimal weight vector. Moreover, it is at the user's disposal to choose which will be the remaining weight, according to the problem. Since it uses the modified one-dimensional bisection method, it requires only that the algebraic signs of the function and gradient values be correct. It is also possible to use this method in training with block of

weights using different remaining weights. In this case, the method can lead to a network training and construction algorithm. This issue is currently under development and we hope to address it in a future communication.

Note that in general the matrix of our reduced system is not symmetric. It is possible to transform it to a symmetric one by using proper perturbations [6]. If the matrix is symmetric and positive definite the optimal weight vector minimizes the objective function. Furthermore, DRTM appears particularly useful when it is difficult to evaluate the gradient values accurately, as well as when the Hessian at the optimum is singular or ill–conditioned [8].

REFERENCES

[1] D. E. Rumelhart and J. L. McClelland eds., *Parallel Distributed Processing : Explorations in the Microstructure of Cognition*, Vol. 1, MIT Press, 1986, pp318–362.

[2] R. L. Watrous, *Learning algorithms for connectionist networks: applied gradient methods of non-linear optimization*, in Proc. IEEE Int. Conf. Neural Networks, San Diego, CA, Vol.2 (1987), pp619–627.

[3] T. N. Grapsa, M. N. Vrahatis, *A dimension–reducing method for solving systems of non–linear equations in* $\mathbb{R}^n$, Int. J. Computer Math., Vol.32 (1990), pp205–216.

[4] J. M. Ortega, W. C. Rheinboldt, *Iterative Solution of Non–linear Equations in Several Variables*, Academic Press, New York, (1970).

[5] J. E. Dennis, R. B. Schnabel, *Numerical methods for unconstrained optimization and non–linear equations*, Prentice–Hall, Englewood Cliffs, NJ, (1983).

[6] T. N. Grapsa, M. N. Vrahatis, *A dimension–reducing method for unconstrained optimization*, J. Comp. Appl. Math. Vol. 66 (1996), pp239–253.

[7] M. N. Vrahatis, *Solving systems of non–linear equations using the non zero value of the topological degree*, ACM Trans. Math. Software, Vol. 14 (1988), pp312–329.

[8] G. D. Magoulas, M. N. Vrahatis, T. N. Grapsa, G. S. Androulakis, *A dimension–reducing training method for feed–forward neural networks*, Tech. Rep. CSL–1095, Department of Electrical & Computer Engineering, University of Patras, (1995).

[9] B. J. Moré , B. S. Garbow, K. E. Hillstrom, *Testing unconstrained optimization*, ACM Trans. Math. Software, Vol. 7 (1981), pp17–41.