Pergamon

Nonlinear Analysis, Theory, Methods & Applications, Vol. 30, No. 7, pp. 4539–4544, 1997
Proc. 2nd World Congress of Nonlinear Analysts
© 1997 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0362-546X/97 $17.00 + 0.00

PII: S0362-546X(96)00368-9

GEOMETRY OF LEARNING: VISUALIZING THE PERFORMANCE OF NEURAL NETWORK SUPERVISED TRAINING METHODS

G.S. ANDROULAKIS[†], G.D. MAGOULAS[‡] and M.N. VRAHATIS[†]

[†]Department of Mathematics, University of Patras, GR-261.10 Patras, Greece; and
[‡]Department of Electrical & Computer Engineering, University of Patras, GR-261.10, Patras, Greece

Key words and phrases: Feed-forward neural networks, supervised training, optimization, back-propagation of error, performance measures, comparisons of training methods, quantitative information.

1. INTRODUCTION

The literature on supervised training of feed-forward neural networks (FNN) [1] includes all algorithms which consider supervised training as the unconstrained minimization of an objective function. The square error over a finite set of input–desired output patterns, containing T representative pairs, is usually taken as the function to be minimized:

$$E(w) = \sum_{t=1}^{T} \sum_{j=1}^{N_L} \left[\sigma \left(\sum_{i=1}^{N_{L-1}} w_{ij}^{L-1,L} y_i^{L-1}(t) + \theta_j^L \right) - d_j(t) \right]^2 , \tag{1.1}$$

where $w_{ij}^{L-1,L}$ is the connection weight from the ith neuron at the $(L-1)$ layer to the jth neuron at the L output layer, y_j^{L-1} denotes the output of the jth neuron belonging to the $(L-1)$ layer, θ_j^L denotes the bias of the jth neuron at the Lth layer, σ is a nonlinear activation function, such as the well known function $\sigma(x) = (1 + e^{-\alpha x})^{-1}$ or the hyperbolic tangent function $\sigma(x) = \tanh(\alpha x)$, and $d_j(t)$ is the desired response of an output neuron at the input pattern t. Each pass through the entire training set to compute E is called an *epoch*. The minimization of $E(w)$ corresponds to updating the weights by epoch, named *batch training*. Training methods originating from the field of numerical analysis such as steepest descent [2,3], nonlinear conjugate gradients [4,5] and second derivative based methods [6,7] have been proposed.

In this framework, the empirical comparison of numerical methods [8] is of great importance in the development of improved training methods and in algorithm selection for problem solving since it is not always evident which algorithm is proper for a given class of applications. Questions concerning "the cost", in terms of function evaluations, the speed of convergence, in terms of epochs and the sensitivity of an algorithm to initial conditions are usually addressed by practitioners of the field. To this end a software package for analyzing and visualizing the convergence behavior of training methods is introduced in this contribution. This package gives quantitative measures for the terms "cost", "fast" and "sensitive". Also it gives pieces of information and displays the geometry of basins of attraction for any training method. It displays also, using different colors, the regions of convergence to the minima for various training methods. Moreover, it indicates the rate of their convergence as well as the region of divergence of these methods. Furthermore, this package gives statistical information regarding the total convergence area in a specific domain for various minima.

2. ACCUMULATING INFORMATION IN A PICTURE

Visualizing the error surface depicting the response of an FNN leads to a better intuitive understanding of the training process. The weights in an FNN can be expressed in vector notation as $w = (w_1, w_2, \dots, w_{n-1}, w_n, w_{n+1}, \dots, w_N)^\top$ defining a point in the N-dimensional real Euclidean space $\mathbb{R}^N$, where N denotes the total number of weights and biases in the network. Thus, geometrically we have to find a global minimum of an objective function with N variables. This function cannot be visualized for $N > 2$. However, using the approach introduced in this section it is possible to "see" in a picture the convergence behavior of a training algorithm.

Each picture's element (pixel) corresponds to an initial (N-dimensional) point of the analyzed training algorithm. Thus, in a network with N weights we can take a finite domain of initial points of the two-dimensional subspace $\mathbb{E}^2$ of $\mathbb{R}^N$ spanned by $\{e^{max}, e^{min}\}$, where e^{max}, e^{min} are the eigenvectors corresponding to the extreme eigenvalues of the Hessian of E at a minimizer w^*, $\nabla^2 E(w^*)$. To be more specific, we apply the algorithms for points

$$w = w^* + \left(c_1 e^{max} + c_2 e^{min}\right), \tag{2.1}$$

where $w \in \mathbb{R}^N$ and c_1,$c_2 \in \mathbb{R}$, by taking a grid of the values of c_1 and c_2 which determine the coordinates of a pixel. Since $\nabla^2 E(w^*)$ is real and symmetric all eigenvalues and eigenvectors are real and the eigenvectors corresponding to distinct eigenvalues are orthogonal. Thus, the values of c_1 and c_2 taken by a grid into an orthogonal parallelepiped represent an orthogonal parallelepiped subset of $\mathbb{E}^2$. The reason for the choice of this two-dimensional subspace is that it reveals useful information explained in the sequel.

Studying the sensitivity of the minimum to small changes to initial points, it is known that, in a sufficiently small neighborhood of w^*, the directions of the principal axes of the elliptical contours (N-dimensional ellipsoids) will be given by the eigenvectors of $\nabla^2 E(w^*)$, while the lengths of the axes will be inversely proportional to the square roots of the corresponding eigenvalues (see also [9]). Thus, a variation along the eigenvector corresponding to the maximum eigenvalue will cause the largest change in E, while the eigenvector corresponding to the minimum eigenvalue gives the least one.

In order to study the behavior of training methods for various directions, including the "extreme ones", we apply the corresponding algorithms for initial points given by Relation (2.1). Of course, it is not necessary to know the coordinates of the minima beforehand, since the package applies an algorithm for any finite domain of initial points and according to its convergence it saves the coordinates of the minima. The minima are distinguished because they are marked with different colors; the first minimum is colored Red, the second Green, the third Cyan, the fourth Gray and so on.

Each picture's element, corresponding to an initial point of the algorithm, takes the color of the minimum to which the algorithm converges. In case where the algorithm does not converge for this initial condition (within an epoch or function evaluation limit) or it converges to a local minimum, the pixel is colored white. Thus, if we have alternate convergence and divergence regions in a picture, then the respective algorithm is not stable since it is sensitive to small perturbations of the initial points. Furthermore, in order to display the rate of convergence (how fast the algorithm converges to a minimum for the specific initial point) we utilize color shades. The dark colors indicate fast convergence while the light ones indicate slow convergence. To this end, we utilize eight different shades per color. In this way, reading a picture we are able to see the regions of rapid convergence.

Moreover, by observing that the colored zones are separated we can easily answer the question whether a minimum attracts the initial points close to it or not. This is a characteristic of the algorithm that we call *"method's reliability"*. Our package counts the total number of the initial points that converge to the closest minimum. Thus, estimating the above percentage, we are able to declare the reliability of the algorithm.

Finally, regarding the computational complexity of an algorithm quantitative measurements are performed and the average of the number of epochs as well as the average of the number of error function evaluations are evaluated.

3. APPLICATION EXAMPLES

The proposed software package has been applied to several problems using FNNs of various dimensions. Here, we exhibit pictures, we give quantitative results and discuss the performance of various training methods in two applications.

3.1. Pattern classification problem: The classification of the four XOR patterns in two classes is a classical problem [2,9,5,3]. The patterns are classified using a simple FNN consisting of $L=3$, two linear, unity-gain input neurons with biases set to zero, and three neurons (two hidden, one output) of logistic activation function with biases to be learned. We give pictures and comparative results for 5 batch training methods: Back Propagation with Variable Stepsize (BPVS) [3], Fletcher–Reeves (FR), Polak–Ribiere (PR), Davidon–Fletcher–Powell (DFP) and Broyden–Fletcher–Goldfarb–Shanno (BFGS) [10]. The termination condition for all algorithms tested was to obtain an error value $E \leq 0.04$ within 1500 error function evaluations. 64000 simulations havė been run in each case with weight vectors initialized according to Relation (2.1).

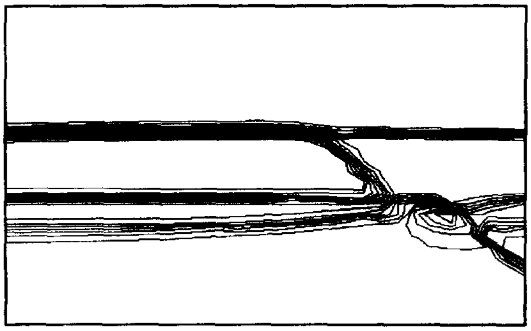
Looking at the pictures of Application 3.1 we easily observe that all methods are sensitive to small changes of the initial weights. Also we observe that FR has the widest white region, while BFGS is the less sensitive to initial weights. Furthermore, the basin of fastest convergence is formed by BPVS since the dark color shade occupies most of the convergence region.

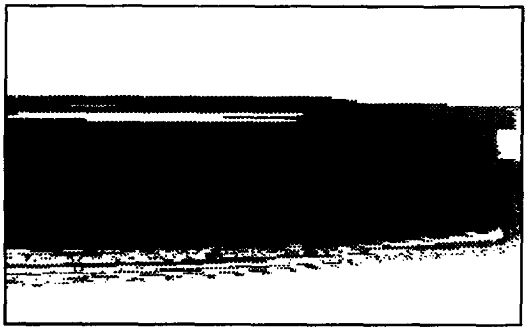
Quantitative results are summarized in Table 1. The first part of the table contains the mean number of error function evaluations *Mn*, the standard deviation *Std*, and minimum/maximum number of function evaluations*Min/Max* that indicates the fastest/slowest run. The reliability *Re* of a method is given in percentage of successive runs. In the second part of Table 1 the percentage of successful XOR simulations per color zone (1–8) starting from the darkest color (fastest convergence speed) is given. According to these results, BPVS is the fastest with an average of 65.2 function evaluations and 96.94% of succesfull runs in color zone 1, while BFGS is the most reliable with 56%. FR has the highest mean number of function evaluations and the lowest reliability.

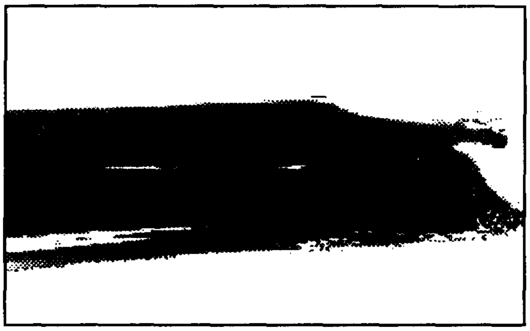
Method	Mn	Std	Min/Max	Re %	1	2	3	4	5	6	7	8
BPVS	65.2	56.3	7/1485	47.6	96.94	2.96	0.03	0.03	0.00	0.01	0.01	0.01
FR	303.8	173.1	20/1498	40.9	20.13	63.80	10.57	2.50	1.22	0.73	0.60	0.44
PR	261.9	108.5	20/1491	48.0	29.81	62.56	6.05	0.66	0.49	0.31	0.08	0.04
DFP	194.3	224.3	20/1499	49.8	73.80	15.60	4.25	2.20	1.64	1.07	0.79	0.66
BFGS	175.1	135.6	20/1473	56.0	71.22	20.67	5.52	2.20	0.27	0.08	0.03	0.00

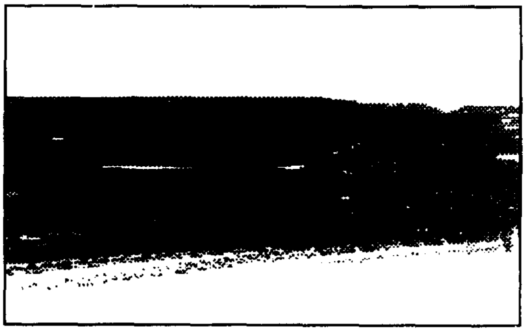
Table 1: Analysis of the convergence behavior of training methods for the classification of the XOR

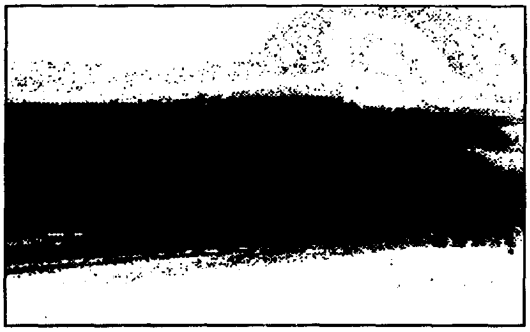
3.2 Continuous function approximation problem: The task is the approximation of a continuous function, $f(x) = \sin(x)$ with points in the interval $(-\pi, \pi)$. The function is learned through a set of 20 input–output pairs, which are uniformly chosen and represent the function adequately (see [11]). These pairs are presented always with the same order to the algorithms. The FNN consists of one linear, unity-gain input neuron no biased, four hidden neurons of hyperbolic tangent activation function and one linear output neuron, all with biases to be learned. We give comparative results for 4 training methods: batch BP with fixed stepsize (BP)[2], Levenberg–Marquardt (LM)[6], Recursive Prediction Error with forgetting factor (RPEF) and Recursive Prediction Error with constant trace

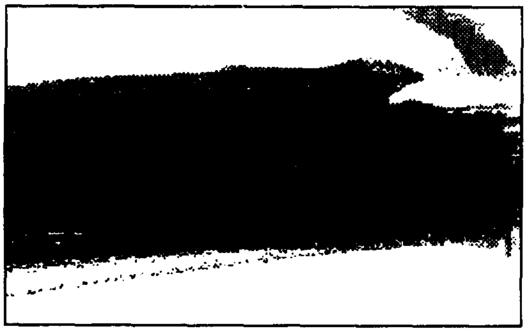

Contour lines


Back-propagation with Variable Stepsize

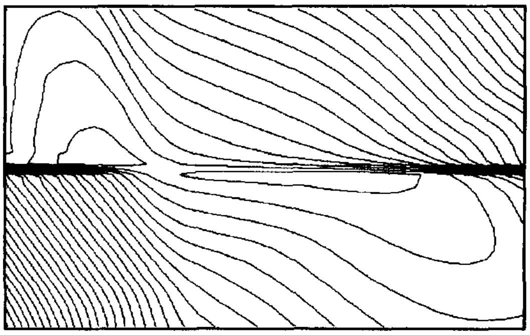

Fletcher - Reeves

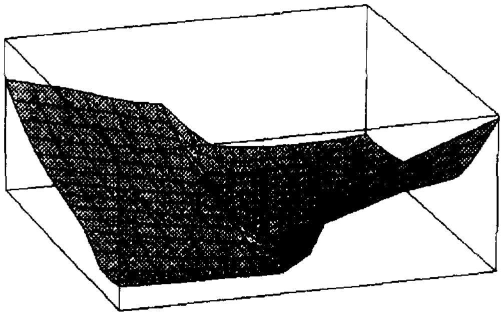

Polak - Ribiere

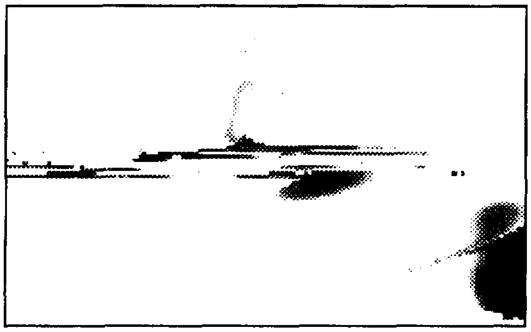

Davidon - Fletcher - Powell

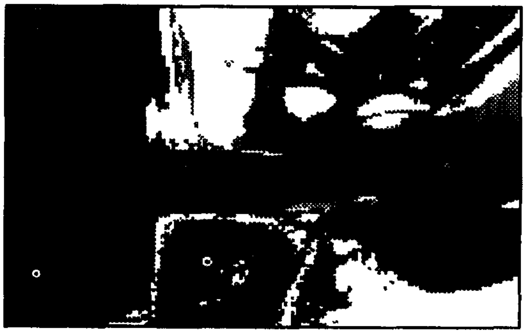

Broyden - Fletchet - Goldfarb - Shanno

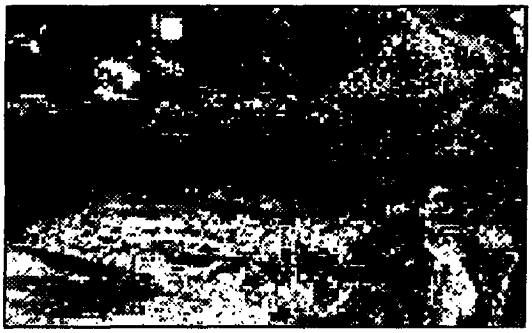
Classification of the XOR Box: $c_1 \in [-50,50]$, $c_2 \in [-90,90]$

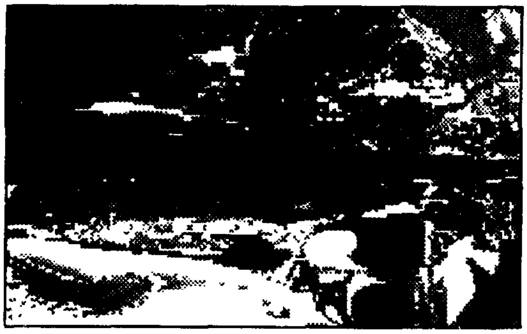

Contour lines


Surface


Back-propagation


Levenberg-Marquardt


Recursive Prediction Error with forgetting factor


Recursive Prediction Error with constant trace

Function Approximation Box: $c_1 \in [-15,15]$, $c_2 \in [-30,30]$

Method	Mn	Std	Min/Max	Re %	1	2	3	4	5	6	7	8
BP	727.7	375.2	12/1500	7.5	5.93	17.10	14.85	16.85	16.01	11.59	9.09	8.51
LM	20.2	15.1	2/199	80.6	83.83	13.14	1.23	1.21	0.26	0.13	0.10	0.08
RPEF	34.9	38.4	1/200	82.6	60.17	17.38	8.91	5.82	3.22	2.01	1.28	1.15
RPEC	40.4	36.5	1/200	77.8	46.57	18.72	17.27	10.45	4.52	1.33	0.69	0.45

Table 2: Analysis of the convergence behavior of training methods for the approximation of the $\sin(x)$

(RPEC) [12]. In this example we compare the algorithms in terms of epochs keeping in mind that BP is notorious for its slow convergence. The goal was to obtain an error value $E \leq 0.01$ within 1500 epochs for the BP and 200 epochs for all other algorithms tested. In all instances 16000 simulations have been run.

From the pictures of Application 3.2 we see that BP has the widest white region although the maximum number of allowed epochs is more than 7 times the one of the other algorithms tested. Also we observe that LM has dark color shades mostly, since the first color shade occupies the highest percentage 83.83% of the total dark area (which occupies the 80.6% of the grid), while RPEF is the most reliable method (82.6%).

4. CONCLUDING REMARKS

The proposed software package can be applied to FFNs of any dimension by taking a finite domain of initial weights of the two dimensional subspace of $\mathbb{R}^N$ spanned by taking the eigenvectors corresponding to the extreme eigenvalues of the Hessian of E at a minimum w^*. This subspace reveals useful information when studying the convergence behavior of various FNN training methods.

REFERENCES

1. HAYKIN S., *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, (1994).
2. RUMELHART D. E., HINTON G. E. & WILLIAMS R. J., Learning internal representations by error propagation, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, 1*, (Edited by D. E. RUMELHART and J. L. MCcLELLAND), pp. 318–362, MIT Press, (1986).
3. MAGOULAS G. D., VRAHATIS M. N. & ANDROULAKIS G. S., Effective backpropagation training with variable stepsize, *Neural Networks*, in press, (1996).
4. JOHANSSON E. M., DOWALA F. U. & GOODMAN D. M., Back-propagation learning for multilayer feed forward neural networks using the conjugate gradient method, *Int. J. of Neural Systems*, **2**, 291–302, (1991).
5. VAN DER SMAGT P. P., Minimisation methods for training feedforward neural networks, *Neural Networks*, **7**, 1–11, (1994).
6. HAGAN M. T. & MENHAJ M. B., Training feedforward networks with the Marquardt algorithm, *IEEE Trans. Neural Networks*, **5**, 989–993, (1994).
7. MAGOULAS G. D., VRAHATIS M. N., GRAPSA T. N. & ANDROULAKIS G. S., Neural network supervised training based on a dimension reducing method, *Annals of Mathematics and Artificial Intelligence*, accepted for publication, (1996).
8. ANDROULAKIS G. S. & VRAHATIS M. N., OPTAC: a portable software package for analyzing and comparing optimization methods by visualizaton, *Journal of Computational and Applied Mathematics*, **72**, 41–62, (1996).
9. JACOBS R. A., Increased rates of convergence through learning rate adaptation, *Neural Networks*, **1**, 295–307, (1988).
10. PRESS W. H., TEUKOLSKY S. A., VETTERLING W. T. & FLANNERY B. P., *Numerical Recipes, The Art of Scientific Computing*, Cambridge University Press, (1992).
11. NØRGAARD M., Neural network based system identification toolbox, Tech. Rep. 95–E–773, Institute of Automation, Technical University of Denmark, (1995).
12. CHEN S., COWAN C. F. N., BILLINGS S. A. & GRANT P. M., A parallel recursive prediction error algorithm for training layered neural network, *Int. J. Control*, **51**, 1215–1228, (1990).