Pergamon

Nonlinear Analysis, Theory, Methods & Applications, Vol. 30, No. 7, pp. 4551–4554, 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)00370-7

ON THE ACCELERATION OF THE BACKPROPAGATION TRAINING METHOD

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

[†]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: Feedforward neural network, Numerical optimization techniques, Steepest descent, Error Backpropagation.

1. INTRODUCTION

Consider a Feedforward Neural Network (FNN) whose lth layer contains N_l neurons, for $l = 2, \ldots, M$. The neurons operate according to the following equation:

$$y_j^l = \sigma\left(\sum_{i=1}^{N_{l-1}} w_{ij}^{l-1,l} y_i^{l-1}\right), \quad j = 1, \ldots, N_l, \tag{1.1}$$

where y_j^l is the output of the jth neuron that belongs to the lth layer, $w_{ij}^{l-1,l}$ denotes a weight from the ith neuron at the $(l-1)$ layer to the jth neuron at the lth layer and $\sigma(x) = (1 + e^{-x})^{-1}$ is the jth's neuron activation function.

The FNN learns to map input patterns to appropriate output patterns. The training is accomplished by modifying the weights of neurons in such a way as to improve the desired mapping between input and output patterns. If there is a fixed, finite set of input–output cases, the square error over the training set which contains P representative cases is:

$$E = \sum_{p=1}^{P} \sum_{j=1}^{N_M} \left(y_{j,p}^M - t_{j,p}\right)^2. \tag{1.2}$$

This equation formulates the objective function to be minimized, in which $t_{j,p}$ specifies the desired response at the jth neuron of the output layer at the input pattern p and $y_{j,p}^M$ is the output at the jth neuron of the output layer M.

To simplify the formulation of the equations throughout the paper we use a unified notation for the weights. Thus, for an FNN with a total of n weights, $\mathbb{R}^n$ is the n–dimensional real space of column weight vectors w with components $w_1, w_2, \ldots, w_n$ and w^* the optimal weight vector with components $w_1^*, w_2^*, \ldots, w_n^*$; $\partial_i E(w)$ denotes the partial derivative of $E(w)$ with respect to the ith variable w_i; $g(w) = \big(g_1(w), \ldots, g_n(w)\big)$ defines the gradient $\nabla E(w)$ of the objective function E at w while $H = [H_{ij}]$ defines the Hessian $\nabla^2 E(w)$ of E at w. Also, throughout this paper $\operatorname{diag}\{e_1, \ldots, e_n\}$ defines the $n \times n$ diagonal matrix with elements $e_1, \ldots, e_n$.

In this paper, we present a new method for dynamically adapting the stepsize in FNN training. This method allows us to give an individual stepsize for each weight. We illustrate its performance in training FNNs for two problems and we compare it with some popular training methods.

2. BACK-PROPAGATION TRAINING AND STEEPEST DESCENT

It is well known that the minimization of the error function E requires a sequence of weight vectors $\left\{w^k\right\}_0^\infty$, where k indicates iterations, which converges to the point w^* that minimizes E.

The most popular supervised training method is the Back-Propagation (BP)[1]. The BP minimizes the objective function E using the following Steepest Descent method (SD) with constant stepsize λ:

$$w^{k+1} = w^k - \lambda g(w^k), \quad k = 0, 1, \dots . \tag{2.1}$$

The optimal value of the stepsize λ depends on the morphology of the error function. The gradient, $g(w)$, is computed by applying the chain rule on the layers of the FNN (see [1]).

The BP approximates a local minimum of E and always converges when λ is chosen to satisfy the relation $\sup \|H(w)\| \leq \lambda^{-1} < \infty$ in some bounded region where the relation $E(w) \leq E(w^0)$ holds for some initial weight vector w^0 [2]. In the neural network implementation, the manipulation of the Hessian is too expensive in computation and storage and the stepsize is usually chosen according to the relation $0 < \lambda < 1$ in such a way that successive steps in weight space do not overshoot the minimum of the error surface. Although BP has proved to be efficient in many applications, it requires a constant stepsize, its convergence tends to be very slow and it often yields suboptimal solutions [3].

A reason why BP may be slow to converge is that a stepsize appropriate in one weight direction is not necessarily appropriate for other weight directions. Furthermore, it may not be appropriate for all portions of the error surface [5].

3. ACCELERATING BACK-PROPAGATION TRAINING BY ADAPTING THE STEPSIZE IN EACH WEIGHT DIRECTION

Several techniques [4,5,6] have been suggested to accelerate training speed. However, these approaches employ heuristic factors and they do not guarantee that the weight updates will converge to a minimizer of E.

In the sequel we briefly present a different approach that eliminates the aforementioned deficiencies. Our method exploits all the local information regarding the direction and the stepsize. The weight updates are based on the $\partial_i E(w)$ and on the following estimates of each weight direction:

$$\Lambda_i^k = |\partial_i E(w^k) - \partial_i E(w^{k-1})| / |w_i^k - w_i^{k-1}|. \tag{3.1}$$

Relation (3.1) can be considered as a local estimation of the Lipschitz constant Λ in the ith direction and its inverse can be used in order to estimate the stepsize in this direction. This means that for a large value of Λ_i a small stepsize is used and vice versa.

As a consequence, we rewrite the BP weight update equation (2.1) in the form:

$$w^{k+1} = w^k - \operatorname{diag}\{\Lambda_1^{-1}, \dots, \Lambda_n^{-1}\}\, g(w^k), \qquad k = 0, 1, \dots . \tag{3.2}$$

Clearly, this iterative scheme coincides with the one-step Jacobi-secant method. For convergence properties of this method see [7,8,9].

In order to speed up training, when we are far from the minimum, the iterative scheme (3.2) can be reformulated as follows:

$$w^{k+1} = w^k - \eta \operatorname{diag}\{\Lambda_1^{-1}, \dots, \Lambda_n^{-1}\}\, g(w^k), \qquad k = 0, 1, \dots , \tag{3.3}$$

where η is a relaxation coefficient. An elegant search technique for adapting η applied to steepest descent method has been proposed in [10]. Following this technique we have to find the smallest positive integer $m_k = 1, 2, \ldots$ for which $\eta_{m_k} = \eta 2^{1-m_k}$ satisfies the following relation:

$$E(w^{k+1}) - E(w^k) \leq -0.5\eta_{m_k} \|\mathrm{diag}\{\Lambda_1^{-1}, \ldots, \Lambda_n^{-1}\} g(w^k)\|^2. \tag{3.4}$$

4. EXPERIMENTAL RESULTS AND CONCLUDING REMARKS

Two experiments have been conducted on training FNNs using BP with the modified dynamic stepsize procedure (3.8), named Multi–Lipschitz Back–Propagation (MLBP).

In the first experiment a 2-2-1 FNN (6 weights, 3 biases) is used to solve the XOR problem [5,1]. The XOR problem is sensitive to initial weights as well as to stepsize variations, and presents a multitude of local minima with certain weight vectors [11]. The weights have been initialized using the Nguyen-Widrow method [12]. MLBP initial stepsize has been taken equal to 4.5. For the BP and the momentum BP (MBP) [5] the following standard values have been chosen: stepsize = 0.75, momentum factor = 0.9. The termination condition is $E \leq 0.04$ within 600 error function evaluations.

In all instances, 1000 simulations have been run and the results are summarized in Table 1. MLBP and MBP are faster than the BP. The aforementioned local minima problem affects the number of successful simulations. A better success has been observed for the MLBP when compared with the BP and the MBP.

Method	Success	Mean	Std
BP	44 %	250.5	96.8
MBP	43 %	240.9	113.1
MLBP	50 %	225.8	143.2

Table 1: Comparison in terms of function evaluations for the XOR problem

The second experiment is on training a 64-6-10 FNN (444 weights, 16 biases) for recognizing numerals from 0 to 9 [13]. The FNN has an 8×8 pixel input and a 10 bit one-hot output representing 0 through 9. The weights have been initialized following a uniform distribution in $(-1, +1)$. The BP fixed stepsize has been set to 0.9; a larger stepsize leads to oscillations. The termination condition is $E \leq 0.01$ and the algorithms have been tested on 100 simulation runs. The heuristics for the ABP [6] have been set as follows: error ratio = 1.04, stepsize increment factor = 1.05, stepsize decrement factor = 0.7.

The results are summarized in Table 2. MLBP is definitely better than the other algorithms escaping shallow local minima and providing fast training.

Method	Initial Stepsize	Success	Mean	Std
BP	0.9	100 %	2174.4	546.6
ABP	1.75	38 %	1870.0	1120.0
	2.00	23 %	2436.0	1116.0
MLBP	1.75	100 %	1907.0	409.3
	2.00	100 %	1886.7	367.5

Table 2: Comparison in terms of function evaluations for the font 8×8 problem

In conclusion, in order to accelerate the convergence speed and to avoid overshooting we propose a method for dynamically adapting an individual stepsize for each weight. This is based on estimates of the local Lipschitz constant that are obtained without additional error function and gradient evaluations. The convergence of the proposed method is guaranteed under suitable assumptions. We also incorporate in the new method a search technique for the determination of the relaxation coefficient η in order to ensure that the value of the error function is sufficiently decreased with every weight update.

REFERENCES

1. 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).
2. GOLDSTEIN. A. A., Cauchy's method of minimization. *Numer. Math.*, **4**, 146–150, (1962).
3. BALDI P. & HORNIK K., Neural networks and principal component analysis: learning from examples and local minima. *Neural Networks*, **2**, 53–58, (1989).
4. CHAN L. W. & F. FALLSIDE F., An adaptive training algorithm for backpropagation networks. *Comput. Speech Language*, **2**, 205–218, (1987).
5. JACOBS R. A., Increased rates of convergence through learning rate adaptation, *Neural Networks*, **1**, 295–307, (1988).
6. VOGL T. P., MANGIS J. K., RIGLER A. K., ZINK W. T. & ALKON D. L., Accelerating the convergence of the back-propagation method, *Biological Cybernetics*, **59**, 257–263, (1988).
7. ORTEGA J. M. & RHEINBOLDT W. C. , *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, (1970).
8. VOIGT R. G, Rates of convergence for a class of iterative procedures, *SIAM J. Numer. Anal.*, **8**, 127–134, (1971).
9. VARGA R., *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, (1962).
10. ARMIJO L., Minimization of function having Lipschitz continous first partial derivatives, *Pacific J. Math.*, **16**, 1–3, (1966).
11. BLUM E. K., Approximation of boolean functions by sigmoidal networks: Part I: XOR and other two variable functions, *Neural Computation*, **1**, 532–540, (1989).
12. NGUYEN D. & WIDROW B, Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights, *IEEE First International Joint Conference on Neural Networks*, **3**, 21–26, (1990).
13. SPERDUTI A. & STARITA A., Speed up learning and network optimization with extended back-propagation, *Neural Networks*, **6**, 365–383, (1993).