Neural Network With Dynamic Optimal Learning Rates and Genetic Algorithm

SENG KAH PHOOI, HAN-LEIH LIU
School of Microelectronics, Griffith University, Kessels Rd, Nathan QLD 4111,AUSTRALIA
S.Phooi@mailbox.gu.edu.au, sikuo@lycos.com

Abstract: - In this paper the authors present a neural network (NN) backpropagation (BP) training with dynamic optimal learning rates and genetic algorithm (GA). An initial learning rate α_0 in the hidden-output layers and proceed to train the NN with the dynamic optimal rates obtained from [1]. By choosing the optimal β_{opt} for each iteration during the training process of the NN, the total squared error J can be found for this initial α_0. The search must then be continued to yield the optimal α_{opt} such that the total squared error J is a minimum. Simulation results have revealed the good performance of the proposed NN BP training with optimal learning rates and GA.

Key-Words: Neural network, backpropagation, genetic algorithm, signal processing

1 Introduction

In recent years there has been a great deal of interest in artificial neural networks and their applications. One of the most popular NNs models is the multilayer network and the related BP training algorithm [3]. BP has been applied to various fields such as signal processing and control engineering. BP is used to adjust the weights of the NN. This algorithm can be considered as gradient descent class algorithm that attempts to minimize the error between the desired and the NN outputs. The weights of the NN are adjusted so that the error is reduced along the descent direction.

One of the major issues in BP is selection of the learning rate in BP. The relatively large or small learning rates may affect the performance of the BP algorithm and may lead to failure of the learning processing. Different authors [1],[2] have investigated the optimal learning rate in BP. Authors in [2] proposed dynamic optimization of the learning rate using derivative information. Nevertheless, the analysis of stable learning rates was not discussed. Authors in [1] have developed the dynamic optimal learning rates of a certain class of fuzzy neural networks (FNNs). However, they only considered the dynamic optimal learning rate in 2- layers NN in certain class of FNNs. They have performed the stability analysis of the learning rate 2-layers NN by minimizing the total squared error between the actual and desired outputs for a set of training vectors. The stable and optimal learning rate, in the sense of maximum error reduction for each iteration during the BP process, can be found in the 2-layers NN.

In this paper, we present the dynamic optimal learning rate and GA for 3-layers NN. The weights of the hidden-output layers of the NN are adjusted using BP with dynamic optimal learning rate proposed in [1]. The optimal learning rate in the input-hidden layers is searched by the GA [4]-[7], which has emerged as a popular family of methods for global optimization. GA performs a search by evolving a population potential solutions through the use of its operators. Simulation example is presented to demonstrate the performance of the proposed method.

The paper is organized as follow. Section 2 briefly describes the 3-layers NN and dynamic optimal learning rate. Section 3 presents how to use GA to tune the learning rate in input-hidden layers. The simulation example is presented in section 4. Finally section 5 concludes the paper with a discussion of the significance of the results.

2 Problem Formulation

A 3-layer NN is established as follows:

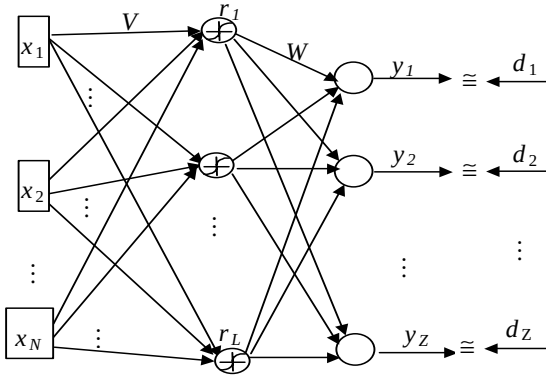

Fig. 1: 3-Layer Neural Network

There are some notations are defined as:

$\underline{r} = [r_1 \quad r_2 \quad \dots \quad r_L]^T \in \Re^L$, training data vector,

$V=[\underline{v}_1 \ \ \underline{v}_2 \ \cdots \ \underline{v}_L] \in \Re^{N\times L}$, weight matrix,

$\underline{v}_i=[v_{1i} \ \ v_{2i} \ \ \ldots \ v_{Ni}]^T \in \Re^N$, i^{th} weigh vector,

$W=[\underline{w}_1 \ \ \underline{w}_2 \ \cdots \ \underline{w}_Z] \in \Re^{L\times Z}$, weight matrix,

$\underline{w}_i=[w_{1i} \ \ w_{2i} \ \ \ldots \ w_{Li}]^T \in \Re^L$, i^{th} weight vector,

$\underline{y}=[y_1 \ \ y_2 \ \ \ldots \ y_Z]^T \in \Re^Z$, actual output vector;

$\underline{d}=[d_1 \ \ d_2 \ \ \ldots \ d_Z]^T \in \Re^Z$, desired vector,

and "T" denotes matrix transpose.

Given P training vectors, there should be P desired output vectors. In matrix notations, we let

$$R=[\underline{r}_1 \ \ \underline{r}_2 \ \cdots \ \underline{r}_P] \in \Re^{L\times P}, \qquad (2.1)$$

the input training matrix,

$$Y=[\underline{y}_1 \ \ \underline{y}_2 \ \cdots \ \underline{y}_P]^T \in \Re^{P\times Z}, \qquad (2.2)$$

the actual output matrix,

$$D=[\underline{d}_1 \ \ \underline{d}_2 \ \cdots \ \underline{d}_P]^T \in \Re^{P\times Z}, \qquad (2.3)$$

the desired output matrix.

Given a set of training vectors, which forms the training matrix R in (2.1), it is desired to use the BP technique to train the above NN so that the actual outputs converge to the desired outputs. The actual output y_z is defined as

$$y_z = \sum_{l=1}^{L} r_l w_{lz} = \underline{r}^T \underline{w}_z \qquad (2.4)$$

r_l, the output of hidden layer (layer II) whose active function is hyperbolic tangent, can be easily computed:

$$r_l = \tanh\Big(\sum_{n=1}^{N} x_{pn} v_{nl}\Big) \qquad (2.5)$$

The actual output matrix Y can be shown as

$$Y= R_{PL}^T W_{LZ} \qquad (2.6)$$

Error function E is defined as

$$E=Y-D = R^TW-D \qquad (2.7)$$

Then we have the total squared error J:

$$J= \frac{1}{2P\cdot Z} TR\,(EE^T) \qquad (2.8)$$

To update weighting factor V and W, the BP method is applied:

$$V(t+1) = V(t) - \alpha_t \frac{\partial J}{\partial V(t)} \qquad (2.9)$$

$$W(t+1) = W(t) - \beta_t \frac{\partial J}{\partial W(t)} \qquad (2.10)$$

$$= W(t) - \beta_t \frac{1}{P\cdot Z} RE$$

Using the chain rule, we get the individual v_{ij} for i=1,…,N, j=1,…,L as:

$$v_{ij}(t+1)=v_{ij}(t)-\alpha_t \frac{\partial J}{\partial v_{ij}(t)}$$

$$=v_{ij}(t)-\alpha_t \frac{1}{PZ}\sum_{p=1}^{P}\sum_{z=1}^{Z}(y_{pz}-d_{pz})w_{jz}(t)\Big(1-\tanh\Big(\sum_{n=1}^{N} x_{pn}v_{nj}(t)\Big)^2\Big)x_{pi}$$

In order to find the optimal learning rate for β_t, we apply the theorem 1 & 2 of dynamic optimal learning rate approach in [1]. For finding optimal learning rate α_t, the GA is used.

3. Tuning Learning Rate Using GA

GAs are iterative search algorithms based on an analogy with the process of natural selection (Darwinism) and evolutionary genetics. The main goal is to search for a solution, which optimizes a user-defined function called the fitness function. To perform this task, it maintains a population or a gene pool of randomly encoded chromosomes (or individuals, solution candidates), $Pop_t = \{\alpha_t^1,\ldots,\alpha_t^{Pop\_size}\}$ for each generation t. Each α_t^i is selected randomly following a uniform distribution over search space and can be binary strings or a real value. It represents a potential solution to the problem at hand and is evaluated. Then, a new population (generation t + 1) is formed by selecting the more fit chromosomes. Some members of the new population undergo transformation by means of genetic operators to form new solutions. After some generations, it is hoped that the best chromosome represents a near-optimal solution.

There are three operators: selection, crossover, and mutation. The selection decides which of the chromosomes in a population are selected for further genetic operations. Each chromosome i in a population is assigned a value φ_i of fitness. The fitness values are used to assign a probability value ρ_i to each chromosome. The probability value ρ_i is defined as

$$\rho_i = \varphi_i \Big/ \sum_{k=1}^{Pop\_size} \varphi_k \ . \qquad (3.1)$$

The chromosome with a larger fitness value has a larger probability of selection. The crossover operation combines the features of two parent chromosomes to form two similar offspring by swapping corresponding segments of the parents. The parameters defining the crossover operation are the probability of crossover Pc and the crossover position. Mutation is a process of occasional alternation of some gene values in a chromosome by a random change with a probability less than the mutation rate Pm.

GAs [7] are used to maximize a function or to do a minimization. In our application, the error function J needs to be scaled and transformed into another function to meet the fitness evaluation requirement. For a given J, $J = \psi 10^{\lambda}$, $1<\psi<10$, the fitness function $\varphi(J)$ is defined as [5]:

$$\varphi(J) = \varphi(\psi 10^{\lambda})$$
$$= \begin{cases} -\lambda + 1 - \dfrac{\psi}{10}, & if\ \lambda < 0 \\ 10^{-(\lambda+1)} + \dfrac{1-\psi/10}{10^{(\lambda+1)}}, & if\ \lambda \geq 0 \end{cases} \quad (3.2)$$

The expression (3.2) finds a larger fitness value for smaller *J*. In other words, if the value of *J* is larger, it will be mapped to a smaller fitness value and vice versa. For example, if *J* is 0.007, then λ = -3 and the equation (3.2) will yield a fitness value of 3.3. If *J* is 10238, then λ = 4 and (3.2) will be mapped to 1.8976e-005.

Following the training process as explained in [1], we start with an initial learning rate α_0 in the hidden-output layer and proceed to train the NN with the dynamic optimal rates obtained from Theorems 1and 2 [1]. By choosing the optimal β_{opt} in each iteration in the training process of the NN, the total squared error *J* can be found for this initial α_0. The search must then be continued to yield the optimal α_{opt} such that the total squared error *J* is a minimum. It is obvious that we only have to search for α_{opt} in the NN. The determination of β_{opt} is from Theorems 1and 2. Otherwise, the NN with two learning rates (to be searched for by GAs, [6]) will require much more searching time.

4 Simulation

Simulations have been performed for adaptive filtering using NN. A 3-layers NN is considered. The input, hidden and output layers consist 10 nodes, 150 nodes and 1 node respectively. The simulation results are shown in Figure 2. This figure illustrates the desired output signal and the NN output with optimal learning rates, β_{opt} and α_{opt}.

For comparison, the same NN is trained with fixed learning rates, α=0.01 and β=0.02. Figure 3 shows the training output of proposed NN with fixed learning rates α=0.01,β=0.02, and desired signal.

The total squared errors of both cases are plotted in Figure 4. Figure 5 illustrates the plots of the weights ||V||+||W|| for the proposed method and the fixed learning rate. The results show the proposed method has faster error and weight parameters convergences. In summary, the simulation results have revealed a good performance of the proposed method compared with the conventional BP with fixed learning rates.

5 Conclusion

This paper has presented a new approach in training NN using BP with dynamic learning rates and GA. The result presented can be used in any dynamic NN that includes simple 3-layers. The optimal learning rate of the hidden-output layers can be obtained from theorem 1 and 2 in [1]. The stability analysis of the dynamic learning rate in the hidden-output layers can also be found in [1]. In this case, we only have to search for the optimal learning rate in the input-hidden layers using GA. Simulation example and performance comparison with the conventional BP with fixed learning rates have revealed the good performance of the proposed method.

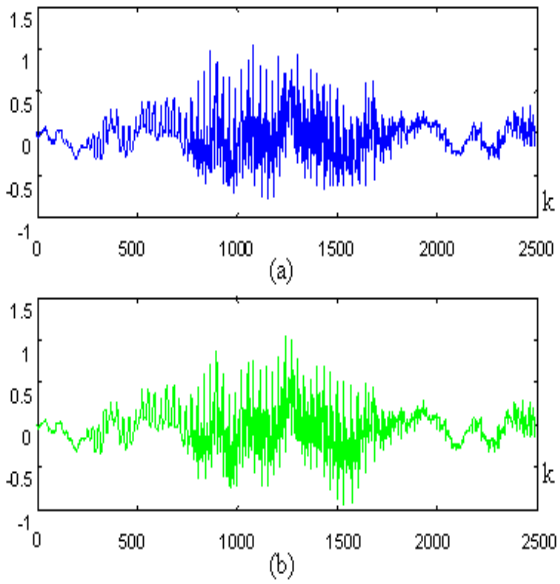

Fig. 2: (a) the training output of the proposed NN with optimal α,β, (b) desired signal.

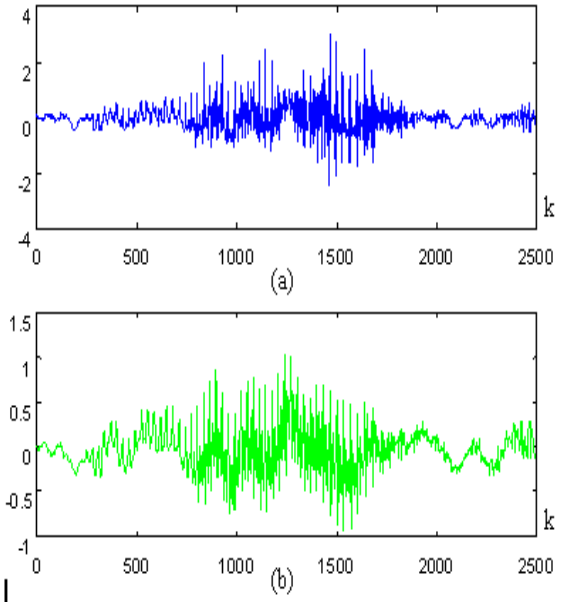

Fig. 3: (a) the training output of the NN with fixed learning rates α=0.01,β=0.02,(b) desired signal

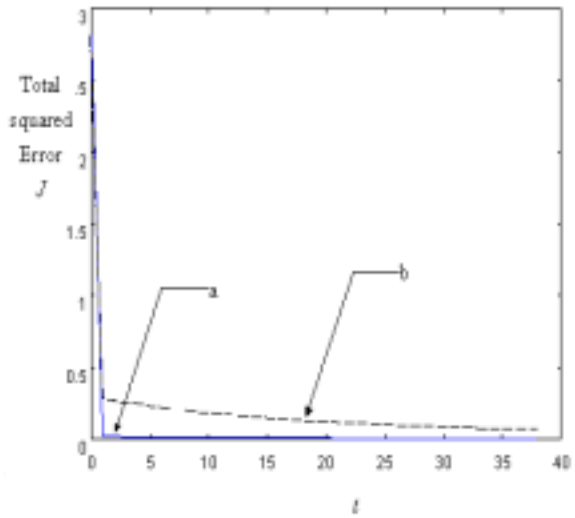

Fig. 4: Performance comparison *Case a*: our optimal learning rates α, β. *Case b*: α=0.01, β=0.02.

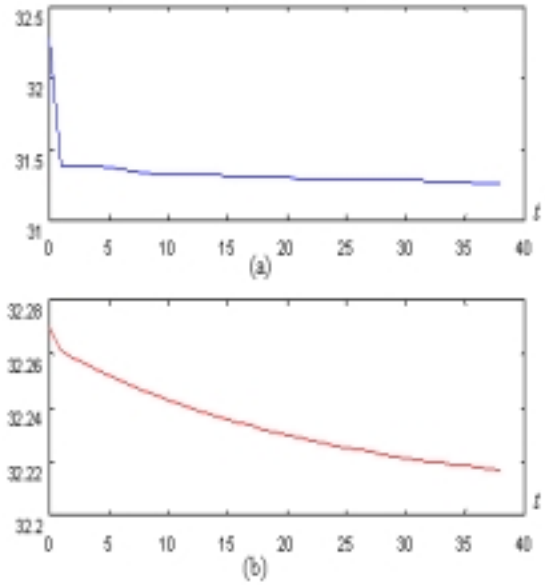

Fig. 5: ||V||+||W||, *Case a*: our optimal learning rates *Case b*: fixed rates α=0.01,β=0.02

References:

[1] C. H. Wang, H. L. Liu, and C. T. Lin "Dynamic Optimal Learning Rates of a Certain Class of Fuzzy Neural Networks and its Applications with Genetic Algorithm", *IEEE Trans. on Syst. Man Cyber*.-part B, Vol. 31, No. 3, June 2001, pp. 467-475.

[2] X. H. Yu, "Dynamic learning rate optimisation of the backpropagation algorithm", IEEE Trans. Neural Networks, vol. 6, pp. 669-677, May 1995.

[3] Rumelhart, D.E. and J. L. McClelland, Parallel Distributed Processing: Explorations in Microstructures of Cognition, Vol. 1: Foundations. MIT Press, Cambridge, MA, 1986.

[4] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, 3rd edition, 1996.

[5] K. S. Tang, K. F. Man, and D. W. Gu, Structured genetic algorithm for robust H^{∞} control systems design," *IEEE Trans. Industrial Electronics* , Vol. 43, No. 5, October 1996, pp. 575-582.

[6] Chen-Chien Hsu, Kai-Ming Tse and Chi-Hsu Wang, Digital redesign of continuous systems with improved suitability using genetic algorithms," *IEE Electronics*., Vol. 33, No. 15, July 1997, pp. 1345-1347.

[7] B. Kosko, *Neural Network and Fuzzy Systems*, Prentice Hall, Eaglewood Cliffs, NJ. 1992.