Factors influencing the choice of a learning rate for a back-propagation neural network

Serge Roy

Defence Research Establishment Valcartier
Command and Control Division
2459, Pie XI Blvd., North (P.O. Box 8800)
Courcellette, Québec, G0A 1R0
email: serge@cc.drev.dnd.ca

ABSTRACT

Neural networks have been used effectively in a number of applications. Most of these applications have used the back-propagation algorithm as the learning algorithm. One of the major problems with this algorithm is that its convergence time is usually very long since the training set must be presented many times to the network. The learning rate has to be selected very carefully. If the learning rate is too low, the network will take longer to converge. On the other hand, if the learning rate is too high, the network may be unstable and may never converge. Up to now, designers of neural network applications had to find an appropriate learning rate for their systems by trial and error. In this paper, a new method to compute dynamically the optimal learning rate is proposed. By using this method, a range of good static learning rates could be found. Furthermore, some studies have been done on factors influencing learning rates.

I. Introduction

Back-propagation neural networks is one of the most widely used supervised training algorithm among neural networks. The back-propagation algorithm requires that the weights of each unit be adjusted so that the total quadratic error between the actual output and the desired output is reduced. A big problem with back-propagation networks is that its convergence time is usually very long.

Selecting a good learning rate reduces training time but can require a lot of trial and error. Trying to find a universal learning rate which fits all needs is unrealistic. Some techniques have previously been developed to find a good learning rate [1],[2] but these methods are usually based on heuristics and do not provide an optimal learning rate. To illustrate what an optimal learning rate is, the total quadratic error versus the learning rate has been computed for weight changes calculated by the back-propagation algorithm (Fig. 1). The point where the total quadratic error is minimal is called the optimal leaning rate. A method has been developed to compute dynamically the near optimal learning rate (NOLR) for back-propagation neural networks [3]. The first part of this paper describes a method to compute the optimal learning rate by using the NOLR method. In the second part of the paper some factors influencing learning rate are studied.

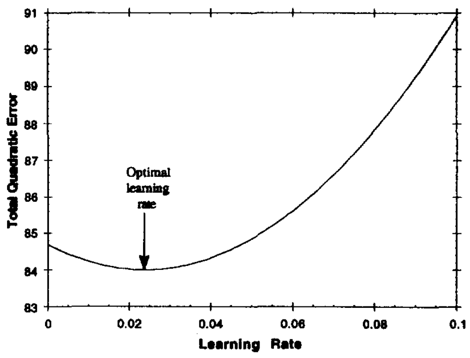

Fig. 1. Quadratic error vs. learning rate

II. Optimal Learning rate

A method was developed to compute the near-optimal learning rate for a two layer back propagation neural network [3].

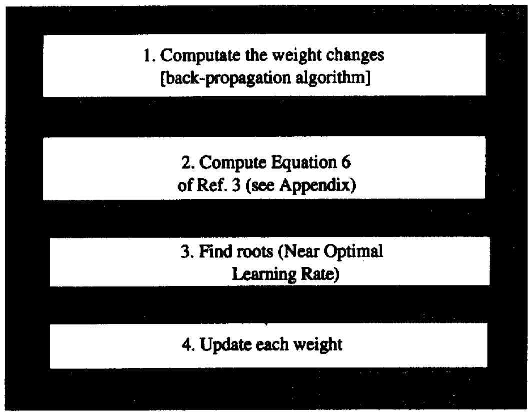
1. Computate the weight changes [back-propagation algorithm]
2. Compute Equation 6 of Ref. 3 (see Appendix)
3. Find roots (Near Optimal Learning Rate)
4. Update each weight

Fig. 2. Near-optimal learning rate algorithm

Fig. 2 shows the algorithm used to compute the near optimal learning rate. The algorithm has four steps: first, the weight changes are calculated by using the back-propagation training algorithm; second, the training set is used again to compute Equation 6 of Ref. 3; third, the resulting cubic equation is solved to find the near optimal learning rate; finally, the weights of each unit are updated using the corresponding weight change and the computed learning rate.

This algorithm has been modified to compute the optimal learning rate. Fig. 3 shows the new algorithm.

The three first steps are the same, but instead of updating weights with the near-optimal learning rate, virtual weights are calculated (Eq. 1) and used to compute a new near-optimal learning rate. The effective learning rate is the sum of all the near optimal learning rates (Eq. 2). As the effective learning rate approaches the optimal learning rate, the near-optimal learning rate goes to 0.

$$w_{virtual} = w_{old} + \eta \Delta w \tag{1}$$

where: $w_{virtual}$ is the virtual weight;
w_{old} is the original weight;
Δw is the weight change calculated by the back-propagation algorithm.

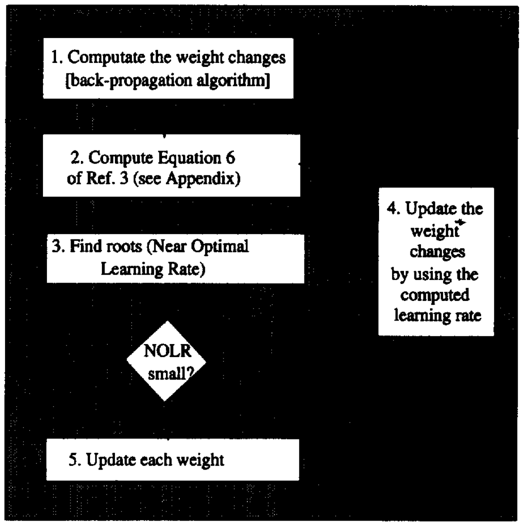
1. Computate the weight changes [back-propagation algorithm]
2. Compute Equation 6 of Ref. 3 (see Appendix)
3. Find roots (Near Optimal Learning Rate)
NOLR small?
4. Update the weight changes by using the computed learning rate
5. Update each weight

Fig. 3. Optimal learning rate algorithm

$$\eta_{new} = \eta_{old} + \eta_1 \tag{2}$$

where: η_1 is the near optimal learning rate;

When the near-optimal learning rate is close to 0.0, weights are being updated by using the optimal learning rate and weight changes calculated by the back-propagation algorithm.

III. RESULTS

This section presents results obtained by computing the optimal learning rate algorithm for the training set shown in Fig. 4. The training set has 201 input-output pairs (67 in each class). The neural network had 2 inputs, 9 hidden nodes and 3 outputs each one representing a class. The algorithm has been implemented in C++ on a HP9000/730 computer.

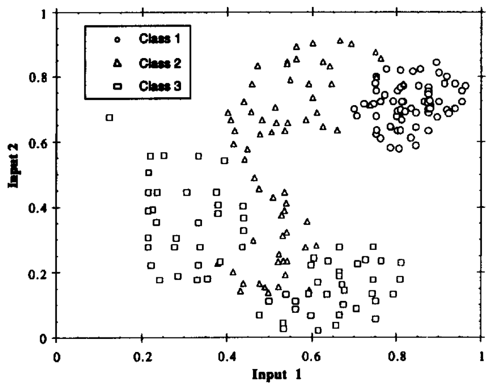

Fig. 4. Training set

Fig. 5 shows an example of the optimal learning rate calculated for one run. The optimal learning rates alternate between two groups of values. Figure 6 shows the distribution of these optimal learning rates. Most of the optimal learning rates are between 0.02 and 0.03. Empirical results show that learning rates between the third and the forth decile are a very good choice for a static learning rate with a momentum of 0.9. In this example, a learning rate of 0.025 would be a very good choice.

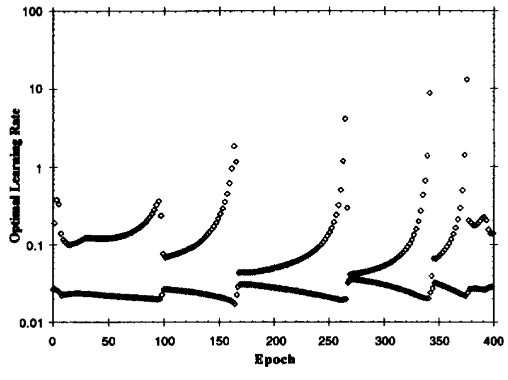

Fig. 5. Optimal learning rate

Fig. 7 shows the cumulative distribution of the optimal learning rate for 10 runs of 400 epochs each (weights were initialized randomly for each run). Three different training sets were used for the simulation.

The optimal learning rates decrease (the curve is shifted to the left) by increasing (same data were used twice) the number of data in the training set.

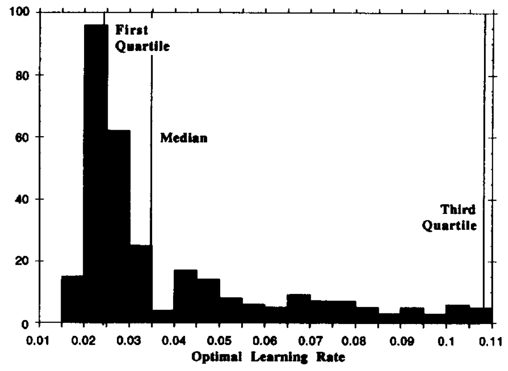

Fig. 6. Optimal learning rate distribution

In order to study the effect of the complexity of the training set on optimal learning rates, an easier training set (Fig. 8) has been used to train the network. The classes of this training set are well divided. As shown in Fig 7, an easier training set has a larger optimal learning rate (curve shift to the right).

As seen in Fig. 7, by increasing the number of nodes in the hidden layer from 9 to 15, the optimal learning rates decrease (the curve is shifted to the left).

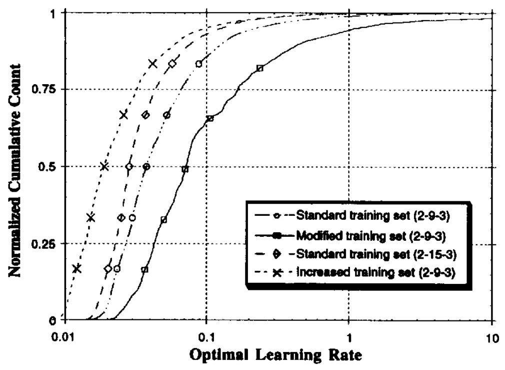

Fig. 7. Cumulative distribution

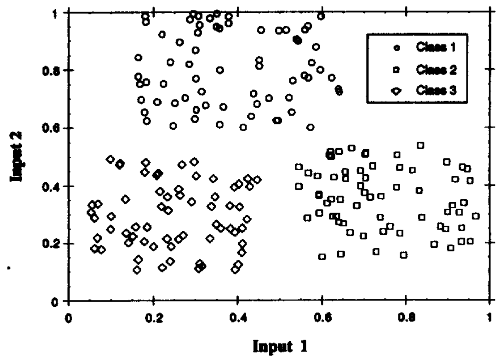

Fig. 8. Simple training set

A good static learning rate for the training set used would be 0.03 (third decile).

Figure 9 shows the total quadratic error for three static learning rates averaged over 5 different runs. As shown, the total quadratic error is lower for the static learning rate of 0.03 than the two other values.

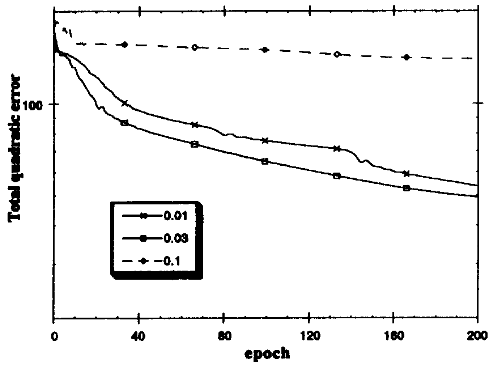

Fig 9. Total quadratic error for static learning rate

IV. CONCLUSION

A number of factors must be taken into account for the choice of a learning rate including the complexity of the training set, the number of examples in the training set and the number of nodes in the network.

A modification to any of these factors involves the selection of a new learning rate. The optimal learning rate algorithm could be used to train a neural network but the algorithm is computationally complex and might take longer to train than a back-propagation neural network having a good learning rate. However, the optimal back-propagation algorithm could be used to select a good learning rate relieving the designer from the frustrating process of selecting a learning rate by trial and error.

V. REFERENCES

1. Janakiraman, J. and Honavar V., *"Adaptive Learning Rate for Increasing Learning Speed in Backpropagation Networks"*, Proceedings of Science of Artificial Neural Networks II, Vol. 1966, pp. 225-235, 1993.

2. Fahlman, S. E., *"An Empirical Study of Learning Speed in Back Propagation Networks"*, Technical Report CMU-CS-88-162, Carnegie Mellon University, 1988.

3. Roy, S., *"Near-Optimal Dynamic Learning Rate for Training Back-Propagation Neural Networks"*, Proceedings of Science of Artificial Neural Networks II, Vol. 1966, pp. 277-283, 1993.

APPENDIX

Equation to compute the near optimal learning rate

$$\sum_{y=1}^{s}\sum_{z=1}^{m}\left[\begin{array}{l}\left(\begin{array}{l}(D_{zy}-O_{zy})\\ -\frac{\eta}{4}\left(\sum_{i=1}^{k+1}\left(C_{iy}v_{iz}+\Delta v_{iz}O'_{iy}+\eta\Delta v_{iz}C_{iy}\right)\right)\end{array}\right)\\ (O_{zy}-O_{zy}{}^{2})\\ \left(\sum_{i=1}^{k+1}\left(\begin{array}{l}4C_{iy}(O'_{iy}-O'_{iy}{}^{2})(v_{iz}+\eta\Delta v_{iz})\\ +\Delta v_{iz}(O'_{iy}+\eta C_{iy})\end{array}\right)\right)\end{array}\right]=0$$

and $$C_{iy}=\frac{1}{4}\sum_{p=1}^{n+1}I_{py}\Delta w_{pi}$$

where:

n	is the number of nodes in the input layer;
k	is the number of nodes in the hidden layer;
m	is the number of nodes in the output layer;

s	is the number of input-output examples in the training set;
O'_{iy}	is the calculated value of node i of the hidden layer for input y of the training set;
η	is the learning rate;
v_{iz}	is the weight value between node i of the hidden layer and node z of the output layer;
Δv_{iz}	is the weight change calculated by the back-propagation algorithm for the weight between the node i of the hidden layer and node z of the output layer;
I_{py}	is the value of cell p of the input layer for unit y of the training set;
Δw_{pi}	is the weight change between cell p of the input layer and cell i of the hidden layer;
w_{pi}	is the value of the weight between cell p of the input layer and cell i of the hidden layer;
D_{zy}	is the desired output of node z for unit y of the training set;
O_{zy}	is the calculated value of node z of the output layer for input y of the training set.