The relation of Weight Update Frequency to Convergence of BP

Jim Torresen[1/2], Shinji Tomita[1], Olav Landsverk[2]

[1]Department of Information Science
Faculty of Engineering, Kyoto University
Yoshida-hon-machi, Sakyo-ku,
Kyoto 606-01, Japan

[2]Department of Computer Systems and Telematics
The Norwegian Institute of Technology,
N-7034 Trondheim, Norway
Email: jimtoer@idt.unit.no

Abstract

This paper proposes a method for estimating the number of iterations required until convergence of a feed-forward neural network for any weight update frequency. The NETtalk neural application has been used to get data on convergence and the best learning rate. Our results show that less frequent weight update implies slower convergence. A formula is derived from the results, describing the relation between the error at start of training to the number of iterations at convergence. Fairly good estimation accuracy is obtained. The estimation formula is of major importance to a parallel BP implementation, which computes different training patterns simultaneously.

1 Introduction

The weight update frequency of Back Propagation [1] has got a major impact on the error convergence during training. However, no theory exactly describes this relation. In this paper we will try to model the relation between the weight update interval (i.e. the number of training patterns that is presented between weight updates) and the error convergence of the BP training algorithm. The convergence of BP deteriorates if the weights are infrequently updated [2]. On the other hand, it is shown in [3] that on a highly parallel computer, it is necessary to exploit *training set* parallelism to obtain a satisfactory performance. That implies that the network weights are updated after a *subset* of the training patterns has been computed. Thus, to be able to utilize todays highly/massively parallel computers in the best way, we ought to know the effect of the weight update frequency to the convergence. This paper considers how to estimate the convergence in advance. The NETtalk network [4] and training set have been used to get experimental results. We derive rules for the best learning rate and the needed training iterations as a function of the weight update interval/the error at start of training.

2 Background

In [5], a heuristic for mapping BP onto a parallel computer is given. This is based on estimating the execution time, $T_{1it}(\mu_i)$, where $1it$ denote one passage through the training set. The estimation is done for K chosen weight update intervals μ_i, $i = 1, \ldots, K$. The μ_i minimizing:

$$T_{total} = T_{1it}(\mu_i)N(\mu_i)$$

is selected. $N(\mu_i)$ is the total number of iterations, needed to obtain convergence, for the weight update intervals μ_i. The paper did not go in depth on the latter issue, which will be examined below.

At least two approaches are possible to determine $N(\mu)$. First, to use existing formulas [2, 6] for the estimation of $N(\mu)$. These are based on specific applications and computes the number of iterations needed based on the learning rate. Thus, in general they will be inaccurate.

A second approach is to run the algorithm for a few iterations (for various weight update intervals) and based on the convergence (error surface) estimate $N(\mu)$. The method benefits from the inclusion of all implemented optimizing techniques in the estimation and quite accurate values of $N(\mu)$ should be obtainable.

In this research, we study the second approach and develop a formula fitting to the error curve.

3 NETtalk experiments

The NETtalk application, proposed by Sejnowski et al in [4], is popular for measuring the performance of parallel BP algorithms. It is a two layer feed-forward neural network that transforms text to phonemes, using 203 input nodes and 26 output nodes. The best performance reported by Sejnowski was by using 120 hidden nodes, which is used in our experiments. To investigate the convergence rate for different weight update intervals, we used the 1000 most common English words (total 5438 characters). The error function E for a training pattern p is given by:

$$E_p(n) = \frac{1}{2}\sum_k (d_{p,k} - y_{p,o,k})^2$$

where n is the iteration number, $d_{p,k}$ is the target output and $y_{p,o,k}$ is the output layer output. We accumulated weight change for a pattern p having $E_p(n) > 0.1$, while a pattern is said be trained when $E_p(n) \leq 0.1$.

For each weight update interval investigated, we searched for the best learning rate (η). For a small μ, the best learning rate was equal to 1.0 and it had to be decreased for larger values of μ. For larger μ, the convergence is more sensitive to the selection of η. The best value for the smoothing term (α) was generally found to be 0.9. Due to the smoothing term, we were able to keep a fairly large learning rate value even with infrequent weight updates. We have not been experimenting with adaptive learning rate during training and other similar methods to speed up the convergence.

This is mainly because, it would have required *much* more computation to make a fair comparison of the convergence for different weight update intervals.

Figure 1 shows the results of the convergence in percentage (called $E_{\%}$) of the characters that have obtained error less than 0.1. Each curve represent one weight update interval. Almost without exception, a larger μ implies slower convergence. The percentage of correctly trained characters went down and stabilized slightly below 5 % wrongly trained characters. This is due to the representation limitation of the network as mentioned in [4]. We regard a network as converged, when $E_{\%} < 5\%$.

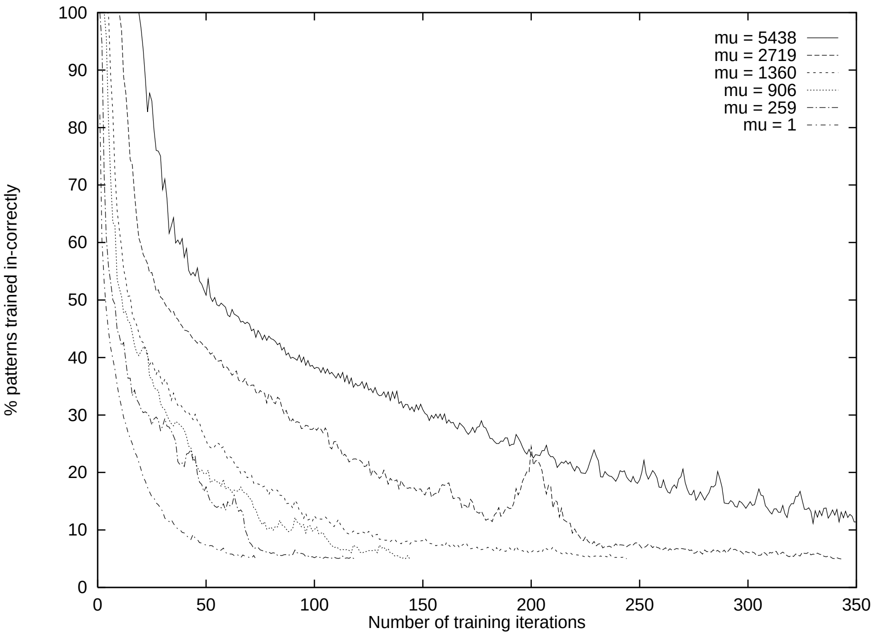

Figure 1: **Percentage of characters that are in-correctly trained for various weight update intervals.**

Figure 2 shows the optimal learning rate for each of the μ_i used in the experiment. Based on these values we establish the following rule, which is comparable to the one in [2]:

$$\eta(\mu) = 7.0(\mu)^{-0.48} \tag{1}$$

Moreover, the number of iterations required for convergence is given in Figure 3. Due to the fairly linear distribution of the points, the least square method is used to establish the experimental rule:

$$N(\mu) = \frac{\mu}{9.6} + 78.2 \tag{2}$$

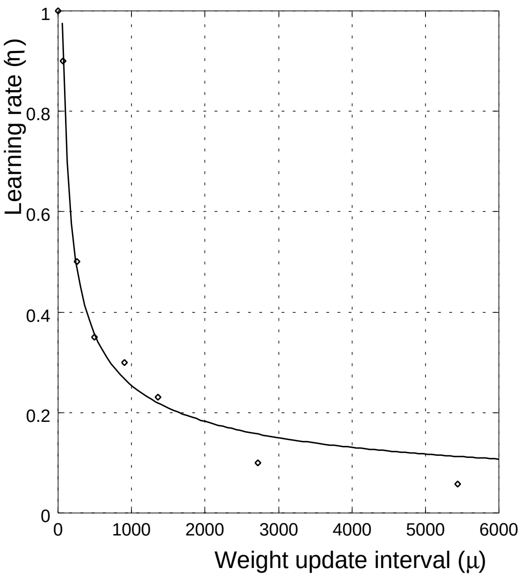

Figure 2: **The best learning rate (η) for each investigated weight update interval. The curve plots the derived rule.**

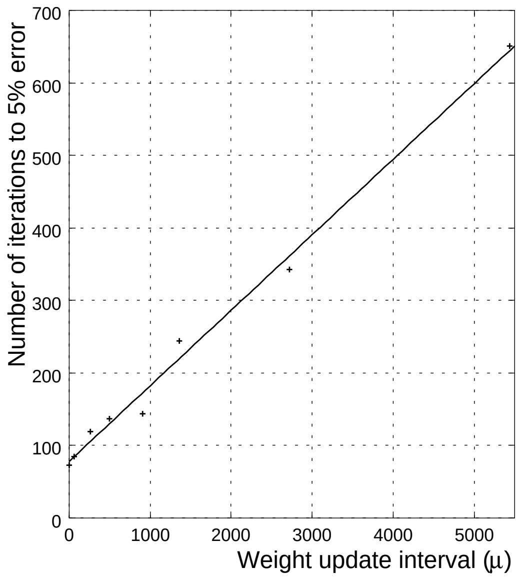

Figure 3: **Number of iterations needed to obtain convergence.**

The linear distribution of points tells us that it is sufficient to estimate the total number of iterations for a few (minimum 2) weight update intervals. These can then be used to find a linear formula for $N(\mu)$.

In the following we intend to show that we can estimate total number of iterations needed based on the error after a small number of training iteration. By looking at the curves in Figure 1, we see that they are of similar form, but different in curvature. To find a curve, fitting to an error curve, logarithmic regression can be used to find an expression, for each μ_i, of the form:

$$E_{\%}(n) = k_1 n^{k_2} \tag{3}$$

where n is the iteration index and k_1 and k_2 are parameters to be estimated. This requires that the points $(\log n_i, \log E_{\%}(n_i))$ are situated on a straight line. Figure 4 shows a logarithmic plot of Figure 1. Above 20 iterations, the curves are becoming more linear for increasing iterations. Thus, we should use regression on this linear part. We made several different regression analysis to search for a curve describing the error convergence. We extended Eq. (3):

$$E_{\%}(n) = k_1 (n + k_3)^{k_2} \tag{4}$$

by k_3, which may be called the offset of the error curve, equal to the number of iterations from start of training till $E_{\%} \leq 95\%$. Equation (4) was solved for the points $(n = 25, E_{\%}(25))$ and (n when $E_{\%}$ becomes less than 5%, 5) with respect to k_1 and k_2, based on registrated values during training. The value of k_2 occured to be fairly constant for all the weight update intervals and averaged to: $k_2 \approx -1.08$. Now we are able to compute k_1 based on *only* the error at $n = 25$ iterations:

$$k_1 = \frac{E_{\%}(n)}{(n + k_3)^{-1.08}} \tag{5}$$

Moreover, the total number of iterations is given by (from (4)):

$$N = \left(\frac{5}{k_1}\right)^{1/-1.08} - k_3 \tag{6}$$

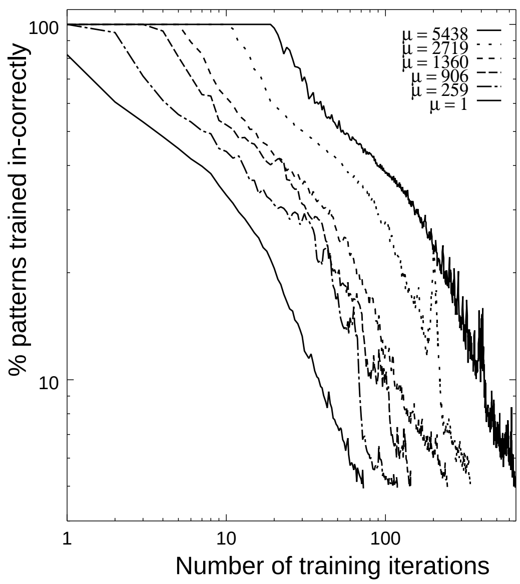

Figure 4: **Convergence of NETtalk, logarithmic plot.**

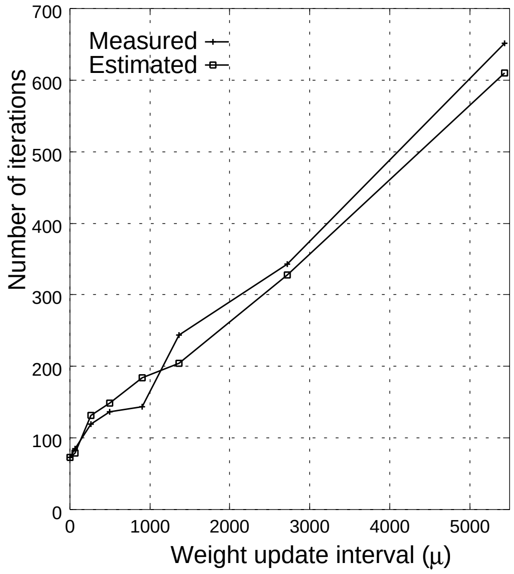

Figure 5: **Comparing measured and estimated values of $N(\mu)$.**

Figure 5 shows both measured and estimated value of $N(\mu)$. The estimation method seems to be accurate, especially for small values of μ_i. As mentioned earlier, it is possible to find a general expression as (2), which may be based on *two* values of (6). To improve accuracy, more μ values should be measured. At the same time, it is beneficial to get results for the μ_i of most current interest.

4 Conclusion

We have proposed a way to estimate the BP convergence for different weight update intervals. The methods differs from other methods in that it is based on parameters measured for the given neural network application. Thus, all speed up techniques involved will be included. The estimation method shows that the approach obtained good accuracy for NETtalk. Verifying the model for other applications is a subject for the future research.

References

[1] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representation by error propagation. In *Parallel Distributed Processing*, volume 1, pages 318–362. The MIT Press, 1986.

[2] Helene Paugam-Moisy. Parallel neural computing based on neural network duplicating. In Ioannis Pitas, editor, *Parallel algorithms for digital image processing, computer vision and neural networks*, chapter 10, pages 305–340. John Wiley & Sons, 1993.

[3] Jim Torresen et al. Parallel back propagation training algorithm for MIMD computer with 2D-torus network. In *Proceedings of International Conference On Neural Network Processing, Seoul, Korea*, volume 1, pages 140–145, October 1994.

[4] Terrence J. Sejnowski and Charles R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, (1):145–168, 1987.

[5] Jim Torresen et al. Optimal mapping heuristic for the back propagation algorithm onto a MIMD computer. In *International Conference on Parallel Processing*, 1995. Submitted.

[6] Akira Sato. An analytical study of the momentum term in a backpropagation algorithm. In Proc. of ICANN–91. In T. Kohonen et al., editors, *Artificial Neural Networks*, volume 1, pages 617–622. Elsevier Science Publishers B.V. (North-Holland), 1991.