Neural Networks, Vol. 6, pp. 719–728, 1993
Printed in the USA. All rights reserved.

0893-6080/93 $6.00 + .00
Copyright © 1993 Pergamon Press Ltd.

ORIGINAL CONTRIBUTION

An Analysis of Premature Saturation in Back Propagation Learning

YOUNGJIK LEE, SANG-HOON OH, AND MYUNG WON KIM

Electronics and Telecommunications Research Institute, Daejeon, Korea

(*Received* 13 *May* 1991; *revised and accepted* 28 *August* 1992)

Abstract—*The back propagation (BP) algorithm is widely used for finding optimum weights of multilayer neural networks in many pattern recognition applications. However, the critical drawbacks of the algorithm are its slow learning speed and convergence to local minima. One of the major reasons for these drawbacks is the "premature saturation" which is a phenomenon that the error of the neural network stays significantly high constant for some period of time during learning. It is known to be caused by an inappropriate set of initial weights. In this paper, the probability of premature saturation at the beginning epoch of learning procedure in the BP algorithm has been derived in terms of the maximum value of initial weights, the number of nodes in each layer, and the maximum slope of the sigmoidal activation function; it has been verified by the Monte Carlo simulation. Using this result, the premature saturation can be avoided with proper initial weight settings.*

Keywords—Premature saturation, Back propagation algorithm, First epoch, Multilayer perceptron.

1. INTRODUCTION

The back propagation (BP) algorithm (Rumelhart, Hinton, & Williams, 1986) is widely used to train multilayer perceptrons (MLP) for many pattern classification problems. Training of MLP is usually accomplished by the iterative update of weights (connection strengths). The weights of the output layer can be changed according to the error between the output value and the target value corresponding to the training pattern, while the weights of the lower layer is updated based on the error back propagated through the weights of the output layer. However, it has two major drawbacks in its learning efficiency (Baba, 1989, 1990; Kollias & Anastassiou, 1989; Jacobs, 1988; Rezgui et al., 1990; Chen & Mars, 1990; Cheung et al., 1990; Hecht-Nielsen, 1989; Li, 1990), such as slow learning speed and convergence to local minima. Convergence to local minima can be caused by the insufficient number of hidden nodes as well as improper initial weight settings. Baba (1989, 1990) tries to solve the latter case using a random optimization method. Slow convergence rate is a common problem of the gradient descent methods, including the BP algorithm. Attempts have been made to speed up learning, among which an adaptive least square algorithm using the second order terms of error for weight updating (Kollias & Anastassiou, 1989) and the BP algorithm with a variable learning rate (Jacobs, 1988) are noteworthy.

Even though an MLP has the sufficient number of hidden nodes, there can be another reason for the drawbacks, namely, "premature saturation." This is a phenomenon in which the error stays almost constant for some period of time during learning. After this period, the error usually decreases to a small value. Rezgui and Tepedelenlioglu (1990) try to adjust the slope of the sigmoidal activation function to solve this problem, and other methods (Chen & Mars, 1990; Cheung, Lustig, & Kornhauser, 1990) also have been suggested.

In this paper, we investigate the premature saturation phenomena to speed up learning. In Section 2, the mechanism of premature saturation is explained using incorrectly saturated output nodes for some training pattern—target value pairs. In Section 3, the probability of incorrect saturation at the beginning epoch of the learning procedure is derived as a function of the range of initial weights, the number of nodes in each layer, and the maximum slope of the sigmoidal activation function. The result shows that the premature saturation can be avoided by setting the proper maximum value of initial weights. This result has been verified by the Monte Carlo simulation for three problems in Section 4, and Section 5 concludes the paper.

Requests for reprints should be sent to Youngjik Lee, Ph.D., Research Department, Electronics and Telecommunications Research Institute, P.O. Box 8, Daedog Science Town, Daejeon, Korea.

2. PREMATURE SATURATION

The premature saturation has been reported by many researchers (Chen & Mars, 1990; Cheung et al., 1990; Rezgui & Tepedelenlioglu, 1990) and it is generally believed that this phenomenon causes slow learning and convergence to local minima. In Rezgui and Tepedelenlioglu (1990), learning is divided into three stages, i.e., an error-convergent stage when the error decreases rapidly in the beginning of learning procedure, a competition stage when the error stays nearly constant for a period of time, and a domination stage when the error decreases rapidly to a small value. Out of these three stages, the period in the competition stage dominates learning time. They argue that the direction of weight update for reducing total error and that for reducing error associated with a specific training pattern compete with each other during this stage, and learning becomes extremely slow. The competition stage corresponds to a special case of the premature saturation in this paper. In some cases, the network stays in the competition stage forever, resulting convergence to a local minimum. In the following, we analyze the cause of the premature saturation.

Consider a single hidden-layer network for a pattern classification application. Here, the range of input values, hidden layer values, output values, and target values is between -1 and 1. Target values have only 1 at the corresponding location and -1 otherwise. The weights are initially set by generating uniform random numbers in a fixed range. When a training pattern is applied to the input of a single hidden-layer network, the output values of the network are calculated through weighted summations and sigmoidal transforms. The error between the output value and the target value corresponding to the training pattern, as well as the slope of the sigmoidal activation function at the weighted sum is involved in weight updating. When the weighted sum to any output node happens to be far from the threshold value of that output node, the slope of the sigmoidal activation function associated with the weighted sum will be very small. In this case, we say that the weighted sum is in the saturation region. When the weighted sum is in the saturation region, the output value will be close to either 1 or -1. If the sign of the corresponding target value is different from that of the output value, we say that the output node for that training pattern is incorrectly saturated. When this happens, the weight change will be very small even though the error is large, and it takes long time to get out of this situation. In any case, output node will be in one of the three regions, i.e., incorrectly saturated, saturated, and unsaturated (active) regions.

At the early stage of learning, output nodes in all three regions exist when any training pattern is applied to the input of a single-hidden layer perceptron. As learning proceeds, the weights associated with unsaturated output nodes change rapidly since the corresponding errors and gradients are relatively large, resulting in the rapid decrease of the total error. Premature saturation can occur when the incorrectly saturated output nodes for some training patterns still remain incorrectly saturated at this point of time. In this case, the chances for weight update happen only when the training patterns causing incorrectly saturated output nodes are presented to the network. However, the amount of weight change is small due to the small gradient of the sigmoidal activation function: This causes the total error remains nearly unchanged.

To calculate the probability of premature saturation, we should know the probability of incorrectly saturated output nodes for all training patterns. During the initial stage of learning, the number of incorrectly saturated output nodes can hardly be changed since the drastic change in weights is necessary for the change. Thus, the probability of incorrect saturation at the beginning stage of learning approximates the probability of premature saturation during learning.

Our argument is supported by an analytic calculation and simulations, which will be described in Sections 3 and 4. We calculate the probability of incorrect saturation at the first learning epoch in terms of the range of initial weights, the number of nodes in the hidden and output layers, the input dimensionality, and the slope of the sigmoidal activation function. Simulation results show close matches to the calculated probabilities.

In the BP algorithm, learning can be classified into two types: the standard type and the batch type (Hecht-Nielsen Neurocomputers, 1980). In the standard type learning, weights are updated for each training pattern, while in the batch type learning, weights are updated once for all training patterns based on the accumulated error. In this paper, an iteration corresponds to the presentation of a training pattern, and an epoch corresponds to the presentation of the entire training patterns. In the next section we analyze the probability of incorrect saturation at the first learning epoch using the batch type learning. However, premature saturation can also be observed in the standard type learning and its analysis is far more difficult.

3. PROBLEM FORMULATION

Let's assume a single hidden-layer perceptron, and let $\mathbf{x} = [x_1, x_2, \ldots, x_N]$ be the input vector, $\mathbf{h} = [h_1, h_2, \ldots, h_H]$ be the value of the hidden nodes, and $\mathbf{o} = [o_1, o_2, \ldots, o_M]$ be the output vector. Also, let $\mathbf{W} = (w_{ij})$ be an $N \times H$ weight matrix between the input and hidden layers, and $\mathbf{V} = (v_{jk})$ be an $H \times M$ weight matrix between the hidden and output layers. Training patterns are denoted here by $\mathbf{x}^{(s)}$, $s = 1, 2, \ldots, p$, and their elements have value the 1 or -1. Suppose that the initial weights are independent, identically distributed (i.i.d.)

uniform random variables between $-w_{\max}$ and $w_{\max}$, and assume that $N \geqq H \geqq M$, which is the case in most pattern classification applications.

Using the notational convention, the weighted sum a_j to the j-th hidden node can be represented as

$$a_j = w_{j0} + \sum_{i=1}^{N} w_{ji} x_i^{(s)} \tag{1}$$

when $\mathbf{x}^{(s)}$ is applied to the input of the network. Here, w_{j0} is the threshold value of the j-th hidden node. From eqn (1), it is easy to calculate

$$E[a_j] = 0, \quad \text{Var}[a_j] = \frac{(N+1)}{3} w_{\max}^2 \triangleq \sigma_a^2 \tag{2}$$

for any j. Since the distribution of w_{ji} is symmetric, $-w_{ji}$ has the same distribution as w_{ji}. In other words, a_j is a sum of $(N+1)$ i.i.d. random variables with finite variances. Thus, if N is large, the central limit theorem (Papoulis, 1984) can be applied to a_j, i.e., a_j can be approximated to a Gaussian random variable with its mean and variance given by eqn (2). Thus, its probability density function (p.d.f.) is given by

$$f_{a_j}(x) = \frac{1}{\sigma_a \sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma_a^2}\right). \tag{3}$$

From eqn (2), it is clear that $a_1, a_2, \ldots, a_H$ have identical Gaussian distributions since they have identical means and variances. Since they are Gaussian random variables for large N, their independency can be implied by their uncorrelatedness, which is described in Appendix A.

The output value of a hidden node is a nonlinear transform of a_j, i.e.,

$$h_j = \frac{2}{1 + \exp(-a_j/T)} - 1. \tag{4}$$

Thus, h_j cannot be a Gaussian random variable, but it is easy to see that its mean is zero since eqn (4) is an odd function of a_j and the p.d.f. of a_j is even. It is clear that the variance of h_j is finite since the range of eqn (4) is finite. Also, h_j and h_k are independent because a_j and a_k are independent.

The weighted sum to an output node can be written as

$$b_k = v_{k0} + \sum_{j=1}^{H} v_{kj} h_j \triangleq v_{k0} + z_k \tag{5}$$

and its mean is zero since $v_{kj}, j = 0, 1, \ldots, H$ has zero mean. The variance of z_k is given by

$$\text{Var}[z_k] = \frac{H}{3} w_{\max}^2 E[h_j^2] \triangleq \sigma_z^2, \tag{6}$$

and the variance of h_j can be calculated using

$$E[h_j^2] = \int_{-\infty}^{\infty} \left(\frac{2}{1 + \exp(-x/T)} - 1\right)^2 f_{a_j}(x)\, dx. \tag{7}$$

The distribution of z_k can be approximated to Gaussian for large H, since (a) $v_{kj}h_j$ and $v_{kl}h_l$ are independent for $j \neq l$, (b) they have the same distributions, and (c) their variances are finite. Thus, for large H, the p.d.f. of z_k can be written as

$$f_{z_k}(z_k) = \frac{1}{\sigma_z \sqrt{2\pi}} \exp\left(\frac{-z_k^2}{2\sigma_z^2}\right), \tag{8}$$

and b_k is a sum of a uniform random variable and a Gaussian random variable.

The k-th output node is incorrectly saturated when b_k lies in the incorrect saturation region of the sigmoidal activation function. Here, a saturation region is assumed to be the region that the slope of the sigmoidal activation function is less than a tenth of the maximum slope (the slope corresponding to $b_k = 0$), i.e., the region that $|b_k| > 3.64T$, where T denotes the temperature of the sigmoidal activation function. If $|b_k| \leqq 3.64T$, we say that the node is in an active region. Based on this assumption, the probability that the k-th output node is incorrectly saturated can be calculated when the training pattern $\mathbf{x}^{(s)}$ is applied to the input of the network. When the target value is 1, incorrect saturation will occur if $b_k \leqq -3.64T$, and when the target value is -1, it will occur if $b_k \geqq 3.64T$. Since the p.d.f. of b_k is symmetric, the probability that a specific output node is incorrectly saturated is given by

$$\Pr\{o_k \text{ is incorrectly saturated}\} = \Pr\{b_k \geqq 3.64T\} \triangleq 1 - q \tag{9}$$

where $q\,(= \Pr\{b_k < 3.64T\})$ represents the probability that a specific output node is not incorrectly saturated. Notice that the probability does not depend on the sign of the target value. Since $b_k = v_{k0} + z_k$, the value of q can be calculated as a double integral in the domain shown in Figure 1, resulting

$$q = \int_{-w_{\max}}^{w_{\max}} \int_{-\infty}^{-v_{k0}+3.64T} f_{v_{k0}}(v_{k0}) f_{z_k}(z_k)\, dz_k\, dv_{k0}$$
$$= \frac{1}{2w_{\max}} \int_{-w_{\max}}^{w_{\max}} G\left(\frac{3.64T - v_{k0}}{\sigma_z}\right) dv_{k0} \tag{10}$$

where $G(x)$ is the Gaussian distribution function, i.e.,

$$G(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-y^2/2}\, dy. \tag{11}$$

To calculate the probability that none of the output nodes is incorrectly saturated, we should see the statistical independency between b_k and b_l. In Appendix B, they are shown to be independent. Thus, the probability that more than one of the output nodes are incorrectly saturated is given by

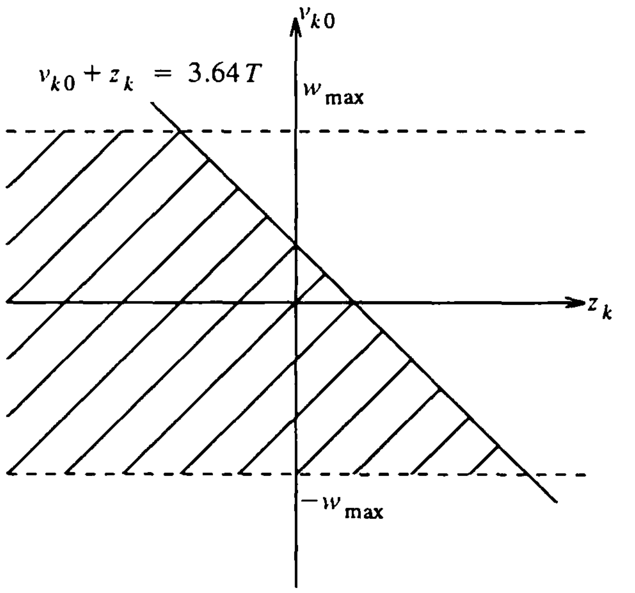


FIGURE 1. The region of integration for computing Pr{ $v_{k0} + z_k < 3.64T$ }.

$$P = 1 - q^M. \tag{12}$$

For computing the probability of incorrect saturation at the first learning epoch, we should consider all the training patterns. Since the training patterns are assumed to be randomly selected from the original population, they can be considered as independent random vectors. Noticing that eqns (2), (10), and (12) do not depend on the distribution of $\mathbf{x}^{(s)}$, we can conclude that the results in eqns (9), (10), and (12) are also true for any training pattern. If we update weights once for each training epoch (the batch type learning), the output values corresponding to $\mathbf{x}^{(s)}$ and $\mathbf{x}^{(t)}$ are statistically independent since the value of b_k is a function of $\mathbf{x}^{(s)}$ only for given $\mathbf{V}$ and $\mathbf{W}$. Thus, the probability of incorrect saturation is given by

$$Q = 1 - q^{Mp}. \tag{13}$$

In Figure 2, the results in eqns (12) and (13) are drawn as a function of w_{max}, H, and T. Figure 2(a) shows that the incorrect saturation at the first learning epoch is more likely to occur when w_{max} is large. In other words, the incorrect saturation can be avoided by choosing an appropriate value of w_{max}. Figure 2(b) shows an interesting result, i.e., the incorrect saturation easily occurs when the number of hidden nodes is large. For MLP to be a universal approximator, it is necessary to have the sufficient number of hidden nodes (Hornik, Stinchcombe, & White, 1989). When H is large and w_{max} is fixed, it is easy to see that q approaches to 0.5 since $\sigma_z \rightarrow \infty$ in eqn (8). In that case, $Q = 1 - (0.5)^{Mp}$. Since p usually is a large integer, the probability of incorrect saturation is close to 1. Thus we should choose very small w_{max} so that the network is safe from the premature saturation. Figure 2(c) is the probability of incorrect saturation in terms of the temperature of the sigmoidal activation function. It is clear that the incorrect saturation rarely occurs when T is large. Rezgui and Tepedelenlioglu (1990) have tried to adapt the slope of the sigmoidal activation function to the squared error in order to escape from the premature saturation, and our result is consistent with his result.

The result derived in eqn (13) is the probability of incorrect saturation at the first learning epoch. As pointed out in Section 2, this value has strong correlation with the probability of premature saturation during learning. It is verified through the Monte Carlo simulations, which will be described in the next section.

4. SIMULATIONS

To support the results derived in the previous section, the Monte Carlo simulations are performed for three problems, i.e., the horizontal-vertical (H-V) line problem (Rumelhart et al., 1986) and the digit recognition problem (Lippmann, 1987). In both simulations, the criterion of premature saturation during learning is as follows: The first 50 epochs can be considered as the initial learning stage and are excluded from checking the premature saturation. After this period, we checked the number of incorrectly saturated output nodes for each epoch. If the number stays nonzero constant integer for 10 epochs, we claim that the MLP is prematurely saturated. If the number decreases in a few epochs, this cannot be considered as premature saturation, since it does not affect the learning speed. Thus, the results of these simulations actually show a relation between the probability of incorrect saturation at the first learning epoch and the probability of premature saturation during learning. For secure verification, 100 simulations have been performed for each test. This number is chosen such that the simulated probability has error less than 10% of the true value with a 95% confidence level.

Another simulation on a handwritten digit recognition problem has been performed to show the relation between the range of the initial weight and the necessary learning epochs. Due to the long computing time (approximately 30 seconds per learning epoch with IBM 3090/200VF), 10 simulations have been performed for each w_{max} in this problem.

4.1. The Horizontal-Vertical Line Problem

This problem is to decide whether an input binary pattern is a vertical line or a horizontal line (Rumelhart et al., 1986). The MLP for solving this problem has 36 (6 × 6) inputs, 14 hidden nodes, and 2 output nodes. Total patterns consist of 6 horizontal line patterns, 6 vertical line patterns, 36 × 12 patterns that have one bit in error, and 630 × 12 patterns that have 2 bits in error. 51 × 12 training patterns are randomly chosen

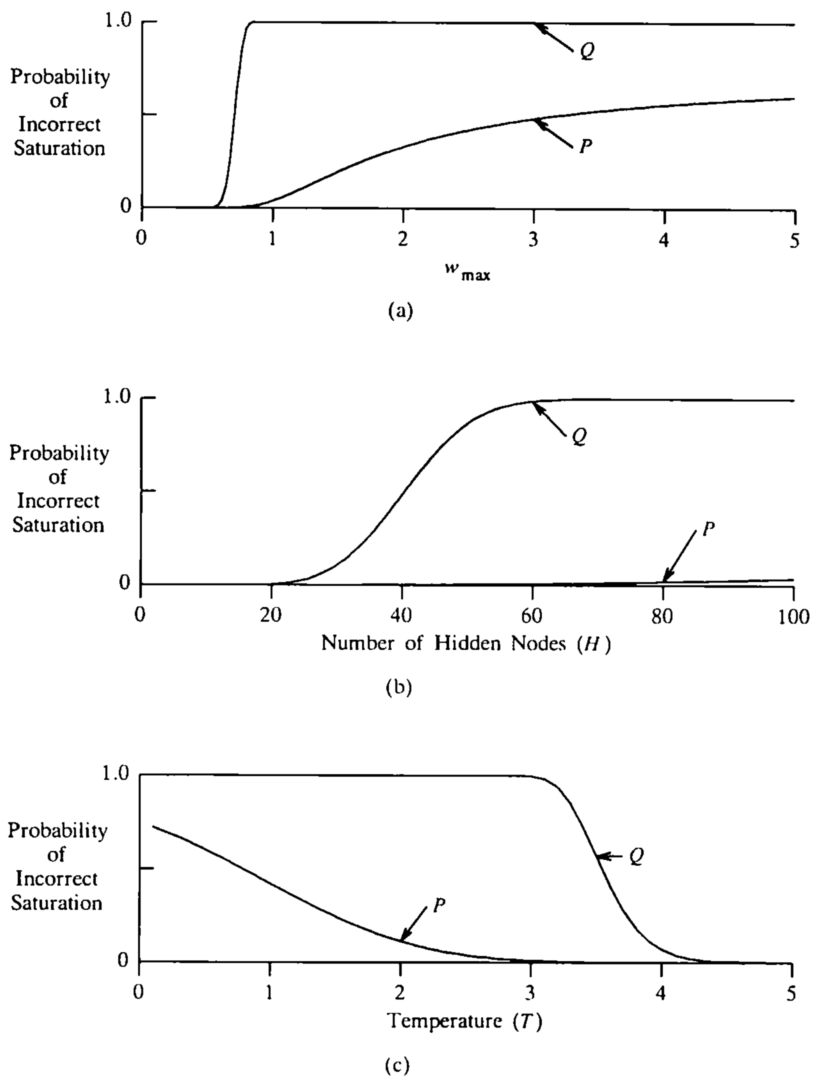


FIGURE 2. The probability of incorrect saturation at the first learning epoch. Here, P and Q corresponds to the probability of incorrect saturation for one training pattern and that for one epoch, respectively. The network was trained with 612 training patterns. (N: the number of input nodes, H: the number of hidden nodes, M: the number of output nodes, T: the temperature of sigmoidal activation function, w_{max}: the maximum value of initial weights) (a) $N = 36$, $H = 14$, $M = 2$, and $T = 1$; (b) $N = 36$, $M = 2$, $T = 1$, and $w_{max} = 0.5$; and (c) $N = 36$, $H = 14$, $M = 2$, and $w_{max} = 2.5$.

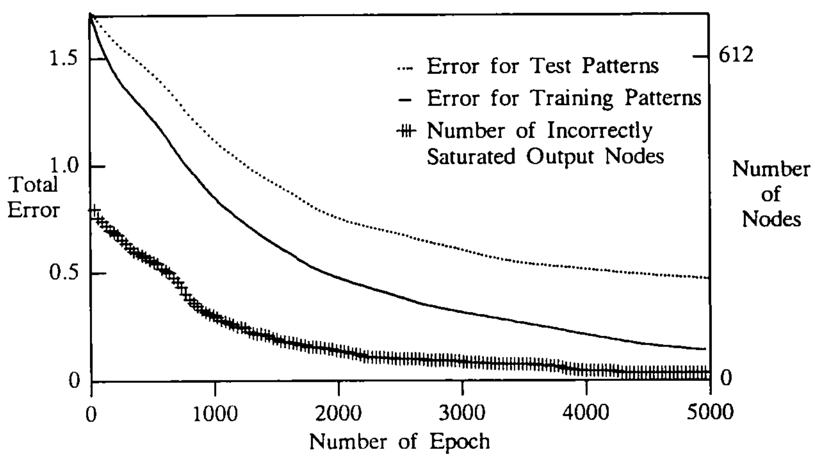
from the total patterns and used to teach the MLP, and another 50 × 12 patterns are randomly chosen for testing the generalization capability of the MLP.

Figure 3 shows an example of the premature saturation. Here, the horizontal and vertical axis denote the number of epochs and the total error, respectively. Two different types of the learning method are used, i.e., the batch type learning (Figure 3(a)) and the standard type learning (Figure 3(b)). Both figures show a strong correlation between the change in the number of incorrectly saturated output nodes and the change of total error.

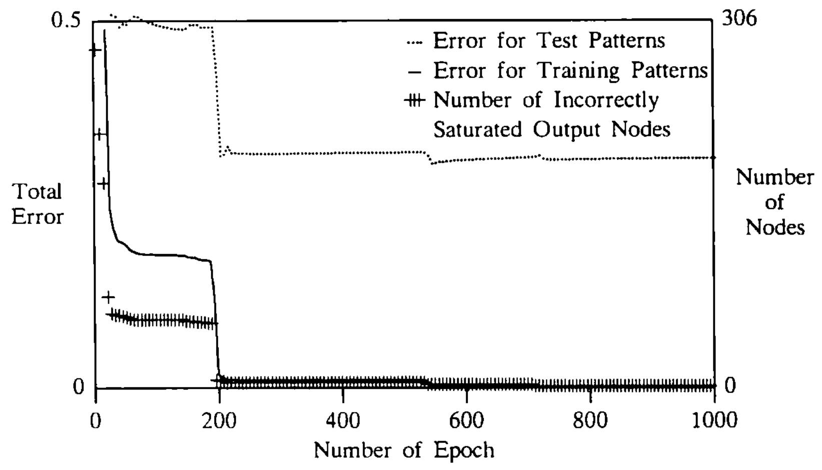
Figure 4 depicts the probabilities of incorrect saturation and premature saturation for this problem. Figure 4(a) shows a result using the batch type learning. It shows a close match between the calculated Q and the simulated Q. Also, we can see a strong correlation between the calculated Q and the simulated probability of premature saturation. Figure 4(b) is a similar result using the standard type learning. Since the theoretical result is not available, only the simulation result is shown. It is clear that the probability of premature saturation increases as w_{max} increases, but the allowable w_{max} is much different from that of the batch type learning. This is mainly due to the fact that a weight update for a specific training pattern introduces serious disturbance to the incorrect saturation of other training patterns.

4.2. The Digit Recognition Problem

Another verification is done with a digit recognition problem. Here, the MLP consists of 120 inputs, 10 hidden nodes, and 8 output nodes. The training patterns



(a)



(b)

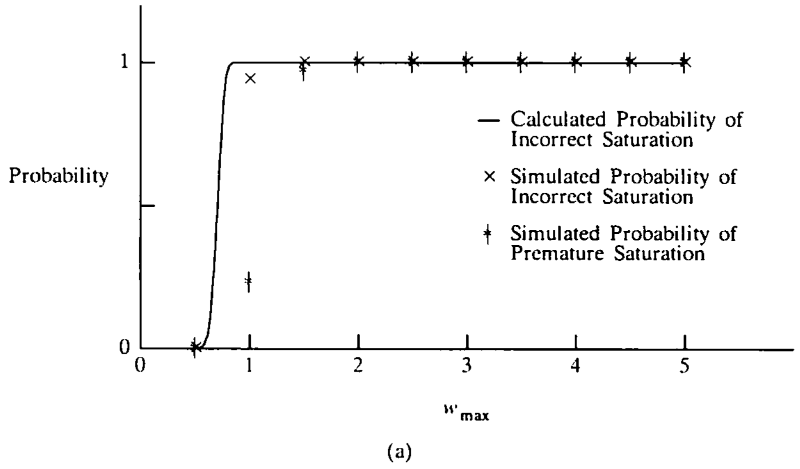
FIGURE 3. Examples of the prematurely saturated error curve for the 6 × 6 H-V problem (w_{max} = 4.0). (a) The batch type learning; and (b) the standard type learning.

are 8 × 3 original patterns (Figure 5(a)) and 8 × 3 × 9 noisy patterns which have less than 6 bits randomly reversed from the original patterns (Figure 5(b)). Notice that the original patterns have three kinds of shift variations. After being trained with the BP algorithm, the network is tested for its generalization capability using another 8 × 3 × 9 noisy patterns. The premature saturation can easily be observed with $w_{max} = 2.5$ in both types of the weight update method.

The probabilities' incorrect saturation and premature saturation in this problem are drawn in Figure 6. In the batch type learning (Figure 6(a)), the calculated result closely matches with the simulation result as well as the probability of premature saturation. In Figure 6(b), the simulated probability of premature saturation using the standard type learning is drawn, in which we can see the similar behavior. Derivation of this probability will be quite a challenging problem.

The learning time of this problem using the standard type learning is analyzed in Figure 7. One hundred trials have been performed to get the probability of successful learning. Here, we assume that a training pattern is correctly classified if the value of the output node corresponding to the training pattern is positive and all the other output values are negative. If all the training patterns are correctly classified within 2000 epochs, we say that the learning is successful. Since the average number of epochs is taken from the set of successful trials, the actual expected number of epochs necessary for successful learning is larger when the probability of success is not 100%. Figure 7 clearly shows that the network learns faster if the range of initial weights is small.

When the range of initial weights is extremely small, more epochs are necessary in the early stage of learning since the amount of the weight change is very small due to the small initial weights. We can see such behavior in the left portion of Figure 7.



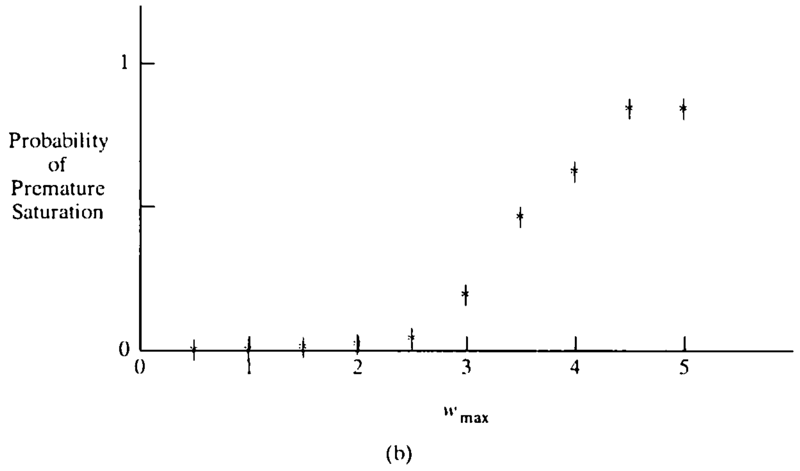


FIGURE 4. The probability of premature saturation in the 6 × 6 H-V problem (N = 36, H = 14, M = 2, T = 1). (a) The batch type learning; and (b) the standard type learning.

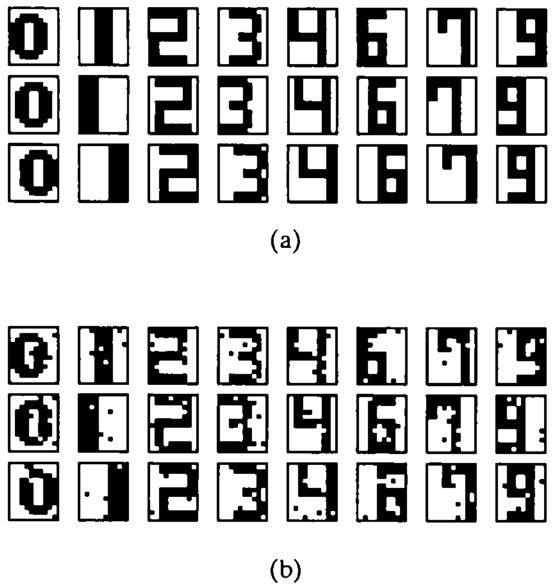


FIGURE 5. Training patterns for the digit recognition problem. (a) Original digit patterns; and (b) digit patterns with less than 6 bits reversed.

4.3. A Handwritten Digit Recognition Problem

In contrast to the previous two problems which are based on the artificially generated data, a handwritten digit recognition problem is chosen to see the effect of the range of initial weights to the learning speed as an example of practical problems. In this problem, 4000 handwritten digit images (10 digits × 10 data sets/person × 40 persons) are collected, in which 2000 images are used for training and the other images are used for testing the generalization capability. One digit image consists of 64 × 64 pixels with 16 levels (4 bits), which is in turn normalized and thinned into 18 × 18 binary image. This image is subdivided into 6 × 6 regions where horizontal/vertical and diagonal/inverse diagonal features are extracted. This 6 × 6 × 2 16 level data is applied to a single hidden-layer network with 72 input channels, 20 hidden nodes, and 10 output nodes.

We assume that the network correctly learns a training pattern if the value of the corresponding output node is greater than 0.8 and the values of the other

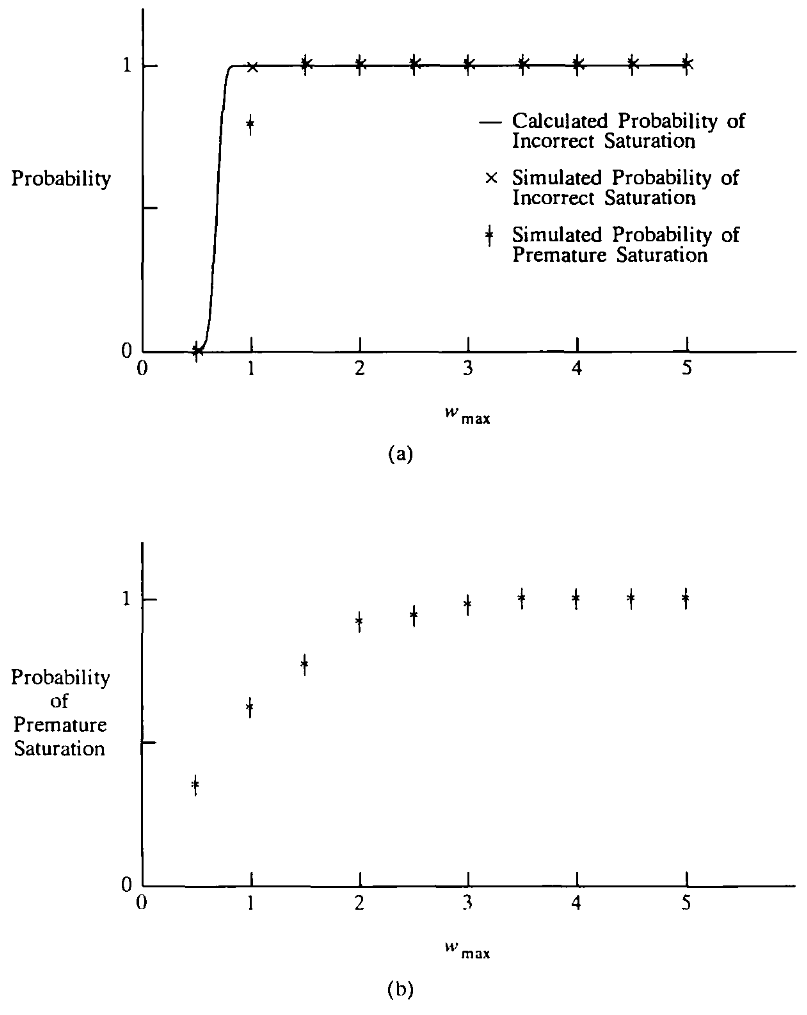


FIGURE 6. The probability of premature saturation in the digit recognition problem (N = 120, H = 10, M = 8, T = 1). (a) The batch type learning; and (b) the standard type learning.

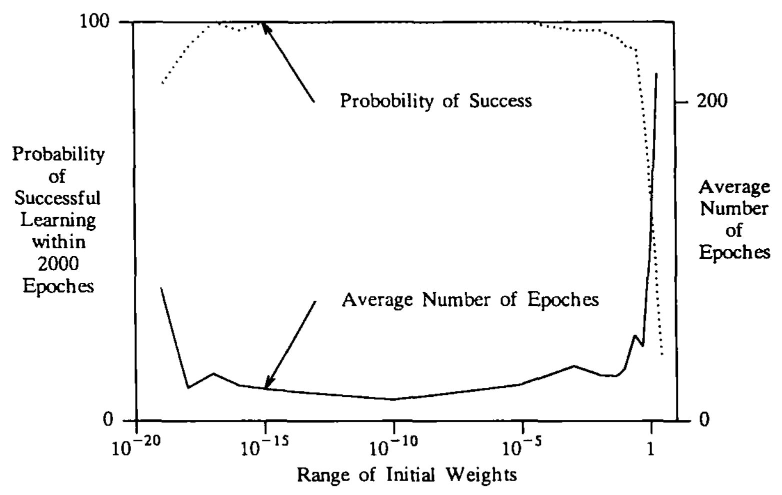


FIGURE 7. Learning time analysis for the digit recognition problem using the standard type learning. One hundred trials have been performed to get the probability of successful learning within 2000 epochs. The average number of epochs was taken from the set of successful trials.

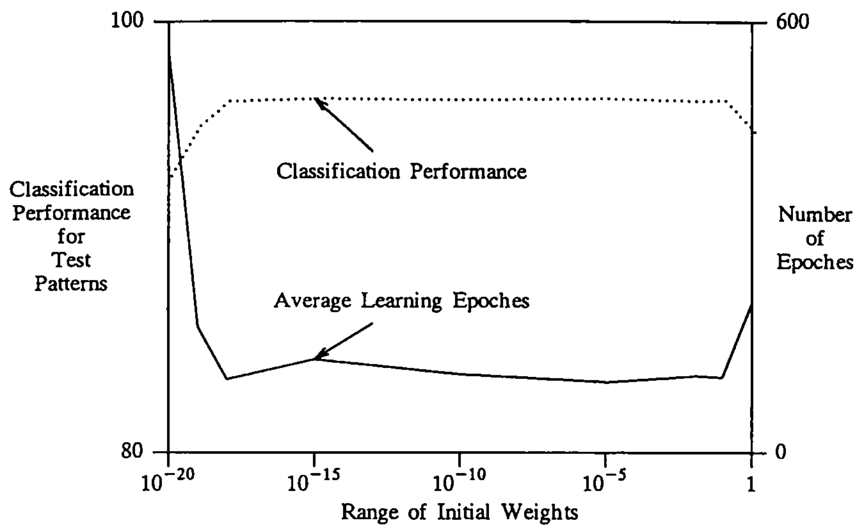


FIGURE 8. Learning time analysis for the handwritten digit recognition problem using the standard type learning. Ten trials have been performed to get the classification performance for test patterns as well as the average learning epoches.

output nodes are less than −0.8. Learning is terminated when the classification performance for training patterns remains constant for 100 epochs. After learning, the generalization capability of the network is examined with the 2000 test patterns. In this examination, we used the max rule, i.e., a test pattern is considered to be correctly classified if the corresponding node has the greatest value among the output nodes. Figure 8 shows the average learning epochs and the classification performances for test patterns in this problem. We can clearly see that smaller initial weight range guarantees faster learning and better classification performance.

5. CONCLUSION

In this paper, the "premature saturation" phenomenon in the MLP learning is analyzed. The probability of incorrect saturation at the first learning epoch is derived as a function of the maximum value of initial weights, the number of nodes in each layer, and the slope of the sigmoidal activation function, which approximates the probability of premature saturation. Results are verified with the 6 × 6 H-V problem and the digit recognition problem. For the batch type learning, the simulated probability of incorrect saturation closely matches with the derived result as well as the probability of premature saturation during learning. For the standard type learning, the simulation result shows similar trends to the batch type learning. We conclude that the premature saturation can be avoided by setting initial weights within a small range for faster training.

REFERENCES

Baba, N. (1989). A new approach for finding the global minimum of error function of neural networks. *Neural Networks*, **2**, 367–373.

Baba, N. (1990). A hybrid algorithm for finding global minimum of error function of neural networks. *Proceedings of International Joint Conference on Neural Networks, Washington D.C.* **I**, 585–588.

Chen, J. R., & Mars, P. (1990). Stepsize variation for accelerating the back propagation algorithm. *Proceedings of International Joint Conference on Neural Networks, Washington D.C.* **I**, 601–604.

Cheung, R. K., Lustig, I., & Kornhauser, A. L. (1990). Relative effectiveness of training set patterns for back propagation. *Proceedings of International Joint Conference on Neural Networks, San Diego*. **I**, 673–678.

Hecht-Nielsen, R. (1989). Theory of backpropagation neural networks. *Proceedings of International Joint Conference on Neural Networks, Washington D.C.* **I**, 593–605.

Hecht-Nielsen Neurocomputers (1989). *HNC Neurosoftware Documents* (pp. 2–8). San Diego, CA: HNC, Inc.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**, 359–366.

Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, **1**, 295–308.

Kollias, S., & Anastassiou, D. (1989). An adaptive least squares algorithm for the efficient training of artificial neural networks. *IEEE Transactions on Circuits and Systems*, **CAS-36**, 1092–1101.

Li, S. (1990). An optimized backpropagation with minimum norm weights. *Proceedings of International Joint Conference on Neural Networks, San Diego* **I**, 697–702.

Lippmann, R. P. (1987, April). An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4–22.

Papoulis, A. (1984). *Probability, random variables, and stochastic Processes* (2nd ed.). New York; McGraw-Hill.

Rezgui, A., & Tepedelenlioglu, N. (1990). The effect of the slope of the activation function on the back propagation algorithm. *Proceedings of International Joint Conference on Neural Networks, Washington D.C.* **I**, 707–710.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representation by error propagation. In D. E. Rumelhart & J. L. McCelland (Eds.), *Parallel distributed processing* (vol. 1, pp. 318–362). Cambridge, MA: MIT Press.

APPENDIX A

To show the uncorrelatedness between a_j and a_k, let

$$a_j = w_{j0} + y_j, \quad a_k = w_{k0} + y_k \tag{A.1}$$

where

$$y_l = \sum_{i=1}^{N} w_{li} x_i^{(s)}, \quad l = j, k. \tag{A.2}$$

Since the initial weights are independent, it is clear that

$$E[y_j y_k] = E[y_j]E[y_k]. \tag{A.3}$$

From eqns (A.1) and (A.3), we can derive that

$$\mathrm{Cov}[a_j a_k] = E[a_j a_k] - E[a_j]E[a_k] = 0. \tag{A.4}$$

APPENDIX B

To show the statistical independency between b_k and b_l, let

$$b_k = v_{k0} + z_k, \quad b_l = v_{l0} + z_l \tag{B.1}$$

where z_k and z_l are defined in eqn (5). Since z_k and z_l are Gaussian random variables as shown in eqn (8), their uncorrelatedness implies their independency. Since v_{kj} has a zero mean and is independent of h_j, it is clear that

$$\mathrm{Cov}[z_k z_l] = 0, \tag{B.2}$$

which implies that z_k and z_l are independent. Since v_{k0} and v_{l0} are independent, we can conclude that b_k and b_l are independent.

NOMENCLATURE

$\mathbf{V} = (v_{kj})$	the weight matrix between hidden and output layer
$\mathbf{W} = (w_{ji})$	the weight matrix between input and hidden layer
$\mathbf{h}$	the hidden node vector
$\mathbf{o}$	the output node vector
$\mathbf{x}$	the input vector
$\mathbf{x}^{(s)}$	the s-th training pattern
$G(x)$	the Gaussian distribution function
H	the number of hidden nodes
M	the number of output nodes
N	input dimensionality
P	the probability that none of the output node is incorrectly saturated
Q	the probability of incorrect saturation in the first learning epoch
T	the temperature of sigmoidal activation function
a_j	the weighted sum to the j-th hidden node
b_k	the weighted sum to the k-th output node
h_j	the output value of the j-th hidden node
p	the number of training patterns
q	the probability that a specific output node is not incorrectly saturated
v_{k0}	the threshold value of the k-th output node
w_{j0}	the threshold value of the j-th hidden node
$w_{\max}$	the range of the initial weights
z_k	the weighted sum to the k-th output node before threshold
σ_a^2	the variance of a_j
σ_z^2	the variance of z_k
$\Delta\mathbf{w}$	the change of weight