Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

THE EFFECT OF TRAINING SIGNAL ERRORS ON NODE LEARNING

Joseph C. Pemberton
Jacques J. Vidal

June 1989
CSD-890041

THE EFFECT OF TRAINING SIGNAL ERRORS ON NODE LEARNING

Joseph C. Pemberton*
Jacques J. Vidal
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024
pemberto@cs.ucla.edu vidal@cs.ucla.edu

Abstract

The response of three node learning rules to errors in the training signal was examined. The discrete (perceptron) learning rule is shown to be very susceptible to target signal errors, and on the average each target signal error causes an output error. In contrast, the linear (Widrow-Hoff) and non-linear (generalized delta rule) learning rules are able to tolerate a large amount of noise in the target signal without effecting the node output function. The ability to tolerate training signal noise is shown to depend on the learning rate and the shape of the non-linear threshold function. For example, a non-linear threshold function, with scale factor equal to 1 and learning rate constant equal to 0.01, produces less than 5% output errors for a rate of 40% errors in the training signal. The training signal error tolerance of linear and non-linear learning rules is explained in terms of the effect of the errors on the weight vector. An example of how the results presented here can be used to characterize the dynamic behavior of a network learning scheme is presented last.

Introduction

This work is motivated by a desire to better understand the effect of node learning rule choice on the behavior of a whole network. Network models are often described as robust. This work is aimed at providing insight into the behavior of node learning rules so that the robustness of a network can be properly attributed to the node learning rule or the network learning procedure. In particular, we show that a single node, that uses a linear of non-linear learning rule, is far more robust than might be expected.

A typical network node consists of input weights (w_{ij}), a threshold (θ_j), an output function (f), and a node learning rule. An extension to the typical node which includes a discrete output function as well as the normal threshold function is shown in figure 1. This addition makes it possible to compare the effect of node learning rules in terms of their effect on the boolean function implemented by the node and thus count the number of times that a node output function is different from the desired output rather than measure the difference between analog outputs. For simplicity, the node inputs and outputs are symmetric, binary signals (i.e. $x_{i\to j} \epsilon \{-1, +1\}$).

The output function for a two input node can be expressed as:

$$x_{j\to} = \begin{cases} +1 & \text{if } w_{1j}x_{1\to j} + w_{2j}x_{2\to j} > \theta_j \\ -1 & \text{otherwise} \end{cases}$$

*Supported in part by Aerojet Electro-Systems under the Aerojet-UCLA Cooperative Research Master Agreement No. D841211, by Nasa NAG 2-302 and by an equipment grant from Hewlett Packard.

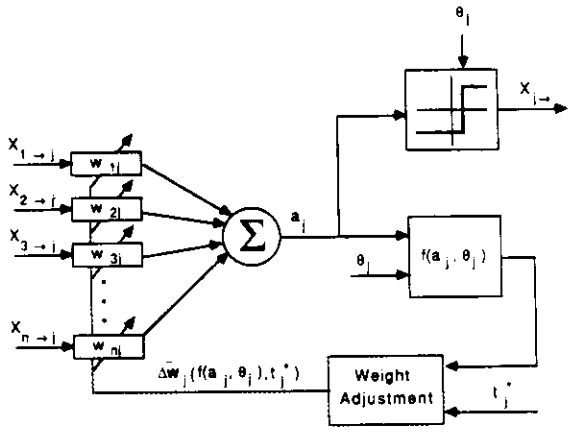

Figure 1: Threshold Logic Unit with Added Discrete Output Function

Node Learning Rules

The three learning rules examined in this paper are the discrete (perceptron) learning rule, the linear (Widrow-Hoff) learning rule, and the non-linear (generalized delta or backward error propagation) learning rule. All three learning rules use the same general format to specify the change to $w_{ij}(t)$:

$$\Delta w_{ij}(t) = \eta(t_j^*(t) - f(a_j(t), \theta_j(t)))x_{i\to j}(t) \quad (1)$$

where η is the learning rate constant, $t_j^*(t)$ is the target or desired output signal (also referred to as the training signal), $x_{i\to j}(t)$ is the input to the node j from node i and $f(a_j(t), \theta_j(t))$ is the node's output feedback signal which is a function of the node activation (a_j) and the threshold (θ_j). For the discrete learning rule, $f()$ is the same as the node's output, whereas for the linear learning rule, the output feedback signal is equal to the node's activation minus the threshold bias ($(a_j(t) - \theta_j(t))$). The output feedback signal for the non-linear rule is more complicated:

$$f(a_j(t), \theta_j(t)) = \frac{2}{1 + e^{s*(-a_j(t)+\theta_j(t))}} - 1 \quad (2)$$

where s is a scaling factor that affects the shape of the (non-linear) threshold function function. The effect of changing the scaling factor is shown in figure 2.

The discrete learning rule only updates the weights when the output differs from the target signal. This means the weight adjustment stops once the weights and threshold implement the desired function. In addition, when the weights are adjusted, each weight is changed by a discrete step equal to $\pm 2\eta$.

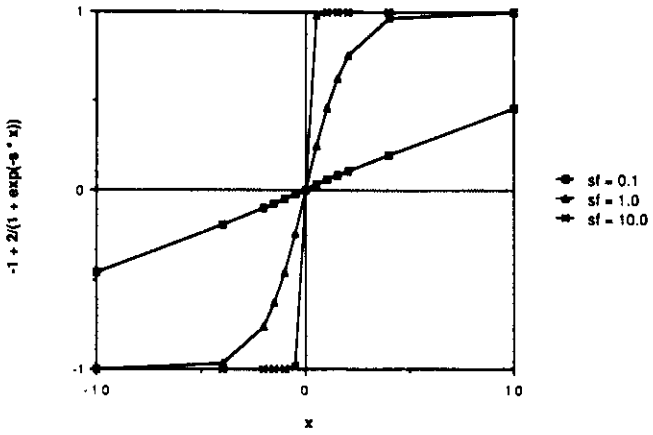

Figure 2: Non-linear Output Feedback Function for Different Values of the Scaling Factor

The linear learning rule updates the weights when the node's threshold adjusted activation differs from the target signal. This means that the weight adjustment continues even after the weights and threshold implement the desired output function[1]. It has been shown [4] that when the statistics of the training set are known, the linear weight adjustment equations can be solved to determine the steady state weight vectors. It has also been shown that the average movement of the weight vector is along a straight line in the weight space toward the steady state location.

The non-linear rule also updates the weights as long as the target signal and the output feedback signal differ. Because the output feedback signal only approaches ± 1 as the node's activation approaches $\mp\infty$, the non-linear learning rule continues to adjust the weights even after the output function matches the target function. For a fixed training set, the weights continue to move toward the center of the ranges of values that will implement the target function [2].

Results

The effect of random errors in the target signal on the output signal was measured for all three learning rules using the following procedure. First the node was trained using an error free training set for 1000 iterations. Next the node was trained with a target signal that was randomly inverted for a fixed percentage of the time as specified by the error rate. After each input pattern and target signal presentation, the weights were adjusted according to learning rule being examined. Output errors were counted each time that the output signal differed from uncorrupted target signal. Trials were conducted on a two input node for each of the 14 linearly separable functions and on a number of linearly separable functions of three inputs for different target signal error rates. The learning rate was kept constant throughout a given trial.

The output error rate as a function of the target error rate for all three learning rules is shown in figure 3. The data is averaged over the 14 linearly separable functions of two inputs. The results clearly show that the linear and non-linear learning rules are able to maintain a consistent output signal when the training set is corrupted. The discrete learning rule, on the other hand, produces output errors roughly as often as it receives target signal errors and is therefore similar to a target signal follower, which just keeps track of the last target signal value, thus automatically producing one output error for each target signal error.

[1]It should be noted that for nodes with more than three inputs the linear learning rule is not able to learn all linearly separable boolean functions.

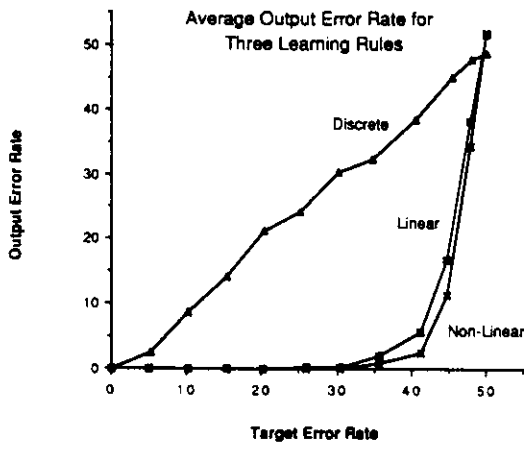

Figure 3: Average Output Error Rate vs. Target Signal Error Rate for Discrete, Linear and Non-Linear Learning Rules

The ability of both the linear and non-linear learning rules to filter out the target signal errors depends on the learning rate constant (η). For the linear learning rule, the effect is shown in figure 4. Notice that as the learning rate increases, the linear node's output error rate approaches that of the discrete node. The learning rate constant value has the same effect on the non-linear node.

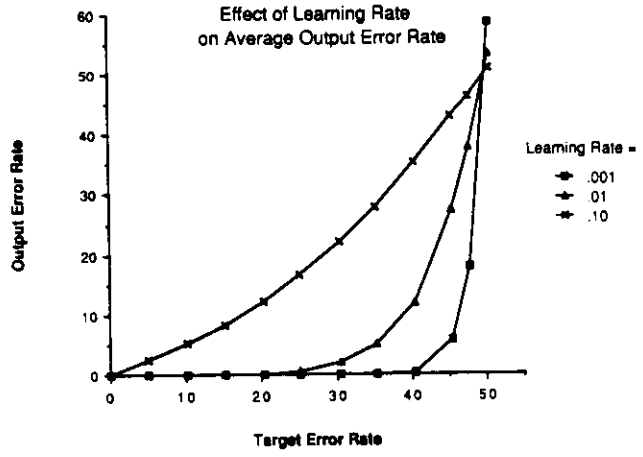

Figure 4: Average Output Error Rate vs. Target Signal Error Rate of Linear Learning Rule for Different Learning Rate Constants

The error filtering behavior of the non-linear node also depends on the shape of the non-linear squashing function used to generate the output feedback signal (see figure 2 and 5). Notice that when the scale factor is small (< 1), the non-linear function resembles the linear output function, and when the scale factor is large (> 10), the function resembles the discrete output function. The training signal error filtering behavior depends on the shape of the output function and therefore the non-linear learning rule can be made to mimic either the linear or discrete learning rules by adjusting the output function scale factor.

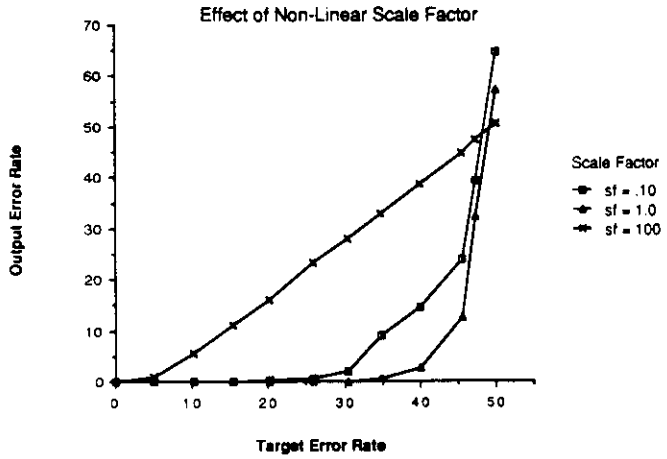

Figure 5: Average Output Error Rate vs. Target Signal Error Rate Non-linear Learning Rule with Different Shaped Output Functions

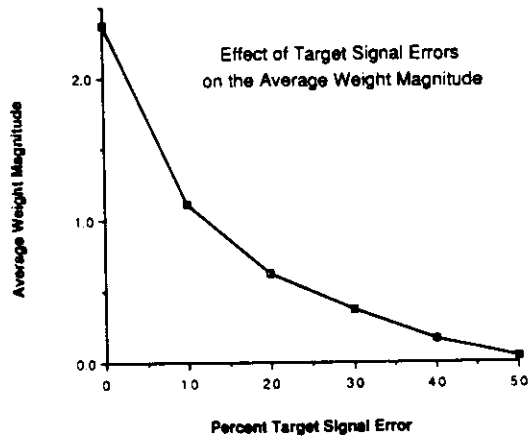

Figure 6: Effect of Random Target Signal Errors on the Weights of a Two Input Non-Linear Node Averaged Over the 14 Linearly Separable Functions of Two Inputs.

Discussion

Empirical evidence has been presented that shows that the linear and non-linear learning rules exhibit an inherent immunity to noise in the target signal. In contrast, the discrete learning rule is very susceptible to random errors in the training signal. This difference in learning behavior is due to the fact that the linear and non-linear learning rules continue to update their weights even though the output function and the target function match. In contrast, the discrete learning rule stops the weight adjustment at the first set of weights that implements the target function. Clearly for a two input node, the linear and non-linear rules produce a set of weights that better represent the target function.

When a node receives a target signal that has been corrupted by random errors, the learning rule updates the weights in the wrong direction. Since the discrete learning rule stops at the first set of good weights, a step in the wrong direction is very likely to change the output function of the node. The discrete learning rule cannot recover from this change until it is discovered as an output error. In addition, the average effect of target signal noise is to reduce the magnitude of the weights. Once the weights are near zero, a single weight adjustment is sufficient to change the output function and cause an output error.

Since the linear and non-linear node learned weights are centered within the ranges of weight values that implement the target function, movement of the weights in the wrong direction not as likely to change the output function because the nearby locations also correspond to the target function. Also, because the weights are adjusted even though the output function and target function match, the next correct target signal will counteract effect of the target error. The net effect of random target signal errors is a reduction in the magnitude of the weights (see figure 6).

Extension to Network Learning

It has been shown that the linear and non-linear learning rules provide a better set of weights for implementing a boolean function when there are target signal errors. These results can easily be extended to input or activation signal errors. The weights produced by the linear and non-linear learning rules are a more reliable way to implement an adaptive logic function, specifically when there is potential for errors in the weight adjustment process signals.

These results can be used to analyze the behavior of different network learning schemes such as competitive learning [7] and backward error propagation [6]. For example, the network learning schemes can be viewed as changing the contents of the training vector in response to network feedback signals. When the network function is first being learned, the node training vectors will resemble a very noisy or random training environment. The result is small weight vectors that are closer to the other possible output functions and are therefore able to adjust more quickly to changes in the long term statistics of the target signal. As the training vectors settle down, the weights will grow to stabilize the node functions. In the end, node function stability translates into network stability.

References

[1] Minsky, M. L. and S. A. Papert, *Perceptrons*. Cambridge: MIT Press, 1969.

[2] Pemberton, J. C., "A Comparison of Continuous and Discrete Learning Through a Geometric Representation," Master's Thesis, Computer Science Department, University of California, Los Angeles, CA, 1988.

[3] Pemberton, J.C. and J. J. Vidal, "Noise Immunity of Generalized Delta Rule Learning," Abstract Proceedings, First International Neural Network Society Conference, Boston, 1988.

[4] Pemberton, J.C. and J. J. Vidal, "When is the Generalized Delta Rule a Learning Rule? A Physical Analogy," Proceedings of the IEEE International Conference on Neural Networks, San Diego, 1988.

[5] Rosenblatt, F., *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Washington, D.C.: Spartan Books, 1961.

[6] Rumelhart, D. E., G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing Volume 1*, ed. D. E. Rumelhart and J. L. McClelland, MIT Press, Cambridge, MA, 1986.

[7] Rumelhart, D. E. and D. Zipser, "Feature Discovery by Competitive Learning," in *Parallel Distributed Processing Volume 1*, ed. D. E. Rumelhart and J. L. McClelland, MIT Press, Cambridge, MA, 1986.

[8] Widrow, G. and M. Hoff, "Adaptive Switching Circuits," Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4, pp. 96–104, 1960.