FAULT TOLERANCE IN ARTIFICIAL NEURAL NETWORKS

Carlo H. Séquin and Reed D. Clay

Computer Science Division
Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720

ABSTRACT

Different strategies for overcoming hardware failures in artificial neural networks are presented. This study focuses on the failure of one or more units in the hidden layer of layered feed-forward networks. First, different types of retraining techniques are investigated, and the required retraining efforts are correlated with the internal representations for specific classification tasks. Subsequently, a practical technique is introduced to achieve true fault tolerance, i.e., to have the network continue to function correctly after failure of one or more hidden units. To achieve this fault-tolerant behavior, hidden units are randomly 'disabled' for some pattern presentations during a standard backpropagation training phase. Prolonged training in this mode can achieve fault tolerance even with respect to fault patterns for which the network is not trained specifically.

This work is supported by a grant from NEC and JSEP.

1. INTRODUCTION

The ability to correctly handle slightly noisy or erroneous input signals is one of the major attractive features of artificial neural networks. This ability is frequently demonstrated by presenting the network with slightly corrupted inputs and noting that the ability of the network to classify these signals has degraded only slightly. A closely related property is the ability of a network to exhibit a certain amount of fault tolerance, i.e., to perform the defined task properly even in the presence of internal failures of its own connections or switching units. This property is not inherent, but has to be built in or has to be acquired through suitable training methods; the latter is the subject of this paper.

Our studies have been carried out in the context of layered feed-forward networks trained with back propagation [1] or with some gradient descent method. The specific architecture on which our experiments have been carried out has full connectivity between the inputs and the units in the hidden layer and also full connectivity between this hidden layer and the units in the output layer. Weights can have positive or negative analog values, and the signals are summed and limited in the switching units with a symmetrical sigmoid function ranging from -1 to +1.

The main type of error that we have concentrated on so far are ''fatalities'' in the hidden layer which are assumed to disable a switching unit and to clamp its output value to zero, i.e., to a neutral value in the middle of the range of possible output values. Such an inoperative unit is assumed to have no further effect on the rest of the network.

The specific task with which we evaluated various approaches to achieving fault tolerance is character recognition. The typical net had 21 inputs arranged in seven ''segments'' of three dots, arranged in the form of the classical seven-segment LED displays (see figure 1). A few different character sets were studied, including the traditional set of 10 digits as well as an artificial set in which all characters had a Hamming distance of three (see figure 2). Normally there were 10 output units, and the number of hidden units was varied from 4 to 14. It was only in this hidden layer that fatalities were introduced and studied. Backpropagation was used with a learning rate varying between 0.125 and 0.05, and a momentum term of 0.9. The initial weights varied randomly between -0.01 and +0.01.

Fig. 1

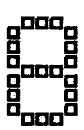

Fig. 2 Characters with Hamming distance of 3

0 1 2 3 4 5 6 7 8 9

2. RETRAINING TECHNIQUES

In a first set of experiments the network was trained in a standard manner. Typically, if a fatality then occurs, the network no longer classifies all inputs properly. If the network has enough structural redundancy, this modified network can normally be retrained to full functionality in less time than it took to train it originally. How much time this retraining takes depends on several factors: the amount of redundancy in the hidden layer, the internal representation of the recognition task, and the retraining method. For the latter variable we distinguished two main cases.

2.1. Simple Retraining

If the network was trained with more than the necessary minimum number of hidden units required to learn its task, we can simply resume training when, due to a fatality, the network ceases to perform properly; similar experiments have also been reported on a Boltzmann machine [2]. The following table shows typical results from our experiments. It lists the number of trials (application of an input/output pair) needed to first train our network with the full set of 14 hidden units (165 trials) and then lists the number of additional trials needed to retrain the network after a succession of fatalities that reduces the number of operational hidden units to only four.

# Hidden Units	14	13	12	11	10	9	8	7	6	5	4
Direct Learning	165	186	202	218	205	215	237	227	257	302	641
Additional Trials	-	14	21	29	30	40	76	73	74	97	340
Total Trials	165	179	200	229	259	299	275	348	422	519	859

The number of trials needed to re-learn the complete task after a unit was killed ranged from 14 to 340 with a systematic trend to larger numbers as the number of functioning hidden units decreases. The total number of trials required to first learn with 14 hidden units and then relearning it 10 times after each subsequent unit has been disabled was 859, compared with 641 trials required to learn the task directly with only 4 hidden units. Thus the network was able to relearn the task several times within a reasonable number of additional trials.

2.2. Retraining with Replacements

An alternative retraining method is to replace the inoperative unit with a new unit with randomly initialized weights. A replacement must be used if the network is left with fewer than the minimum number of hidden units required to learn its task. This method could be implemented with a suitable software/hardware environment containing a pool of reserved units to replace damaged units. It obviously requires some self-check mechanism that indicates which unit has failed. The results for two experiments, on networks with four and seven hidden units, respectively, are shown in the table below. It lists the average number of trials to train the network with the given number of hidden units and also lists the average number of trials it takes to retrain the network after reseting the weights associated with any one of the hidden units.

# of Hidden Units	4	7
Average # Training Trials	641	227
Average # Retraining Trials	270	92

This is about the same amount of retraining that was required without replacing the damaged hidden units (of course this is only true up to the minimum number required). Thus it appears that adding a few redundant hidden units beyond what is absolutely necessary to perform a given task provides an effective basis for repairing a faulty network without the need for any physical reconfiguration of the architecture.

3. TRAINING FOR FATALITY TOLERANCE

In a second set of experiments we studied training techniques that specifically aimed at creating robustness with respect to fatalities of hidden units. The goal is to bring the network into a state in which it continues to perform correctly even when one or more hidden units are disabled.

The modification to the training procedure consisted in disabling one or more of the 14 hidden units at random for each trial. Thus the network was trained to minimize the error for all these cases as well as for the fault-free state.

3.1. Multiple Temporary Fatalities

We first investigated the extra effort required to train the net for all specific patterns of one, two, or three fatalities. During the training for each pattern, one, two, or three of the hidden units were randomly disabled. The following table shows the results of training the network to correctly classify the inputs with any combination of up to three of the hidden units being disabled.

# of Fatalities	0	1	2	3
# of Trials	286	676	2066	15,118(*)

(*) At 6,000 trials, only 0.8% errors (28/3640).

3.2. Single Temporary Fatalities

In more recent experiments, we studied to what degree training for fault tolerance against single failures leads to a more robust representation that will enable the network to withstand even more severe fatality combinations than those that it was trained for specifically. The training here involves just one hidden unit being randomly disabled for each trial. The following table shows the surprising result that the network can become robust to even 2 and 3 random fatalities, given enough additional training.

Trained with Single Fatality						
Number of trials	500	1000	5000	10,000	30,000	100,000
Test 1 fatality	94%	100%	100%	100%	100%	100%
Test 2 fatalities	62%	85%	99%	100%	100%	100%
Test 3 fatalities	30%	52%	82%	89%	95%	99%

How easily this ''extended'' fault tolerance can be achieved depends on the way the character set is defined and represented in the hidden layer. A character set with an uneven separation in weight space between its individual representatives will lead to longer retraining times than an evenly distributed set of characters with a certain minimum Hamming distance between any pair of representatives.

We are currently also investigating the role of artificially injected noise with the goal to enhance the robustness of the internal representation and potentially produce extended fault tolerance within a shorter training period.

4. CONCLUSION

We have investigated various techniques to achieve fault tolerance in artificial neural networks. Retraining techniques can be used successfully and the damaged units do not need to be replaced. Another practical technique consists in randomly disabling hidden units during the training phase in order to produce an internal representation that can perform the given task correctly even with some of these units disabled. To our surprise, we found that with prolonged training the network can achieve fault tolerance even with respect to more severe fault patterns for which it had not been trained specifically.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," Parallel Distributed Processing: Explorations in the Microstructure of Cognition, eds. D. E. Rumelhart, J. L. McClelland, and PDP Research Group, Bradford Books, Cambridge, MA, 1986.

[2] G. E. Hinton and T. J. Sejnowski, "Learning and Relearning in Boltzmann Machines," Parallel Distributed Processing: Explorations in the Microstructure of Cognition, eds. D. E. Rumelhart, J. L. McClelland, and PDP Research Group, Bradford Books, Cambridge, MA, 1986.