ISSN: 2348 - 2117

Available Online at www.ijetcr.org

International Journal of Engineering Technology and Computer Research (IJETCR) 2 (1) 2014, 01-04

FAULT TOLERANT MODELING FOR ARTIFICIAL NEURAL NETWORKS ARCHITECTURE

***Jitendra Joshi, Nisha Singh, Reeta Chainani**

Jayoti Vidyapeeth Women's University, Jaipur, Rajasthan, India

Received 30 December 2013; Accepted 25 January 2014

ABSTRACT

Artificial Neural Networks are composed of a large number of simple computational units operating in parallel they have the potential to provide fault tolerance. One extremely motivating possessions of genetic neural networks of the additional urbanized human body and other animal is their open-mindedness against injure or destroyed to individual neurons. In the case of biological neural networks a solution tolerant to loss of neurons has a high priority since a graceful degradation of performance is very important to the survival of the organism. We propose a simple modification of the training procedure commonly used with the Back-Propagation algorithm in order to increase the tolerance of the feed forward multi-layered ANN to internal hardware failures such as the loss of hidden units.

Key Word: ANN, Fault, Tolerant, Real, MLP 0976

1. INTRODUCTION:

One extremely motivating possessions of genetic neural networks of the additional urbanized human body and other animal is their open-mindedness against injure or destroyed to individual neurons. It is supposed, this is a significance of the circulated representation used by the nervous systems of human body. The big number of neurons and interconnections to each other available makes it easier to obtain such a distributed representation by exploiting redundancy, as was earlier realized by von Neumann [1]. However, the presence of a large number of neurons and interconnections is not enough to guarantee such fault tolerance. The learning mechanism has to be able to exploit the capability of the available hardware according to its priorities.

In the case of biological neural networks a solution tolerant to loss of neurons has a high precedence since graceful dreadful conditions of performance is very important to the survival of the organism. It is still unclear how living organisms achieve such fault tolerance since not enough is known about their learning mechanisms. For artificially created systems, based on artificial neural networks or not, such a graceful degradation in relation to damage of its internal components is highly desirable since it directly reduces maintenance costs and increases overall productivity.

In this, we propose a simple adjustment of the training procedure commonly used with the Back-Propagation algorithm in order to increase the tolerance of the feed forward multi-layered ANN to internal hardware failures such as the loss of hidden units. The proposed method switches, during training, between the different possible fault configurations, i.e. all possible configurations are also trained and forced to share the same set of network weights.

2. Approaches of fault tolerance modeling:

The conventional approach to reach fault tolerance has been to duplicate several times the system performing an important task. If it is simple to detect when the system is faulty inside the required time stage, one could basically have immediately one of the duplicates performance until it becomes faulty. Then it is simply replaced by one of its non-faulty copies. In this scheme only one of the copies is performance at a given time stage. This is known as standby redundancy [2]. In other cases it may be complicated to make a decision, if the system is faulty surrounded by the necessary time stage. Different one in such belongings is to have all copies performance at the same time stage accepting the equal input signals. Every one copy develops by its own output separately of the other copies. In the general system output can then be produced by using a selection procedure, such as a mass conclusion or decision.

The difficulty with the conventional approach is that it can be costly to produce the duplicates, i.e. the tradeoff between cost and redundancy. Although it seems always

*Corresponding author: Jitendra Joshi | Email: lect.jitendra29@gmail.com

necessary to have some being without a job in order to have some quantity of fault tolerance, it may be probable to exploit the being without a job in more efficient behavior. In this paper review, we aspire to exploit the additional that exists in artificial neural networks in terms of a large number of units and weights to make them fault tolerant.

Single effectual way to exploit additional, assuming that we are handling with composite tasks, is to partition the task into sub-tasks and to have numerous sub-systems. Suppose that it is probably to complete such partition in such a way that there is no one-to one communication among the sub-tasks and sub-systems. Every one sub-system contributes to a number of sub-tasks and each sub-task is completed by a number of sub-systems.

In this case, if the number of sub-systems and sub-tasks is large sufficient and no sub system is vital to any of the sub-tasks, the loss of a relatively small set of sub-systems randomly selected will probably affect the performance of all sub-tasks.

The main complexity with such an approach is to devise the task decomposition into sub-tasks and the division of the power of each sub-system to each sub-task.
As model of the above strategy imagines that the task is to paint a square board with dimensions 5m-by-5m and there are 50 painters available. The standby redundancy approach would use only one painter at a time until he becomes *"faulty".* Then he is replaced by another one and so on. Alternatively we can divide the task into 25 sub-tasks where each sub-task is to paint a 1m-by-1m allocated part of the board. The 50 painters then go around and paint just a small portion of a large number of the 1m by 1m squares. The loss of 25 randomly selected painters probably will affect all squares but only a small portion of each square will be left unpainted, that is the loss of painters is uniformly distributed over the 25 squares, the sub-tasks.

3. Fault Tolerance with Neural Networks Architecture:
Artificial Neural Networks are collected of a large number of simple computational units operating in similar way, they have the possible to provide fault tolerance. However just the presence of a big number of units cannot assurance that the ANN will be fault tolerant. The learning algorithm has to be capable to exploit or organize the existing excess of units in such a way that the network is fault tolerant, if fault tolerance is one of our requirements. In other words, the learning algorithm is responsible for decayed the task into the proper sub-tasks and divides the computational power of the units among such sub-tasks consequently.

Today the most popular algorithm used to train a feed forward multi-layered artificial neural network is the back-propagation algorithm. The back-propagation algorithm, by using a incline exploring procedure, tries to reduce an error function defined as the squared output error averaged over the set of training data. No priority, however, is given to the network fault tolerance. For a unique network topology number of hidden layers, number of hidden units in every one layer, it is very probable that there are numerous solutions sets of weights which will have similar satisfactory squared output errors over the set of training data but different degrees of tolerance to hardware failures.

Normally it is assumed that the effect of such duplication would be only to improve the fault tolerance of the network. Astonishingly, they found out that feed forward ANN with such duplication, the simplest form of redundancy learn considerably faster than unduplicated ones. They also showed that feedback ANN with such duplication will also come together quicker to their solution. They quarrel that duplicated feed forward and feedback networks also necessitate less accurateness in inter-unit communication, i.e. the weights can be specified with less accuracy.

The fault tolerance matter of feed forward neural networks as a nonlinear constrained optimization problem by important the concept of a maximally fault tolerant neural network. The difficulty of training a maximally fault tolerant neural network is defined as finding a set of weights that performs the required mapping according to a set of input-desired output training patterns with the added constraint that when any of the hidden units is removed the mapping error measured over the set of training patterns should not increase by more than a user-defined parameter ⍰⍰> 0.
A problem with the above formulation arises if there is even just one hidden unit which causes a large increase in the mapping error while all the other units cause a small increase. Depending on the problem in hand, too much importance will be placed on just one unit, making the problem more difficult than it needs to be.

4. Fault Tolerance Techniques for Artificial Neural Networks:
A process for improving the operational fault tolerance of MLP's by injecting a single fault during each training epoch [3]. In their simulations, the effect of the fault injected was to group the output of a hidden unit to zero. They found that such training produced a MLP network which would face to multiple faults. They concluded that this was due to a more robust internal representation being created. A similar method in which constraints

were placed on the MLP's fault tolerance, and then a set of weights estimated solving the problem by using a large-scale constrained nonlinear programming technique [4]. They note that by using fault tolerance as a constraint, better generalization is obtained.

The Function networks, they found that the results obtained were much more marked in GRBF's than in MLP's [5]. If during each training epoch many units were faulted in the GRBF network, it is found that the process of the final trained network would not degrade even after about 10% of the network's weights deleted. They also examined initializing the weights in the MLP in a "carefully selected manner" as occurs in GRBF's. It is found that this did improve the fault tolerance of the MLP, though only slightly.

Improving the fault tolerance of layered feed forward neural networks solving function approximation problems [6]. Gaussian hidden units were used which implies that they only respond over a limited volume of the input space. Their method confines the highest output of every one hidden unit to a little value which very much reduces its contribution to the final output, although this does imply that a large amount of hidden units will be required. Every one output is formed from the preservative response of several units, and this implies that a degree of redundancy exists. However, their simulation does not take account of the enlarged level of faults that will occur in such a neural network due its enlarged amount and difficulty. For this reason, it is uncertain whether this method does actually go ahead to improved fault tolerance.

An alternative approach to improving the fault tolerance of MLP's has been undertaken by who consider extending the error function used by the back-error propagation learning algorithm to include information on the "robustness" of the network [7]. The term they add to the error function is:

$$E_{ym} = \frac{1}{2N_h} \sum_p \sum_i \sum_k \left(y_i - y_{i,k} \right)^2$$

Where $\mathbf{y_i}$ is the actual output, $\mathbf{y_{i,k}}$ is the output with node **k** faulted. These measures the normalized enlarge in error due to the effect of all probable single faults in all hidden nodes and for all patterns. An MLP network with 10 hidden units has then trained on the ***"XOR"*** function using the modified back-error propagation algorithm. It is found that fault tolerances have enlarged, but the solution found is not ideal, resulting in reduced accuracy. They suggest that this may have been due to the MLP being attentive in a local minimum. Prater and Morley also examined this method, though they used a conjugate training technique, and found that it did give consistent results [8]. This may be due to the better optimization properties of conjugate incline over that of back-error propagation.

Another approach considered by Bugmann et al was to locate the unit which would cause the maximum damage when removed, and to replace another unit with a copy of it. The pruning method employed was to select the hidden unit which had the largest weight on its output. A similar approach has also been taken again by Prater and Morley [8]. Bugmann et al found that the final trained MLP is *"robust",* and its accuracy greater than that resulting from the first method described above. However, their results are very limited. Prater and Morley considered both larger networks and more realistic problem domains, and they concluded that this technique gave very inconsistent results.

Another approach which involves adjusting the bias of units, these results from their observation that a fault causes both information loss and a bias change [9]. The technique involves storing the average input excitation for every hidden and output unit. When a fault occurs, biases are altered such that the displaced excitation values are restored to their stored values. They found that this is very effective for improving reliability of result, though it is clearly unproductive next to faults occurring in an output unit's bias weight values.

5. Fault Tolerance of "Real" Neural Networks:

It is very important to examine the fault tolerance of neural network models at the theoretical or conceptual level, but deliberation must be also be made of how to implement a neural network model using some fabrication technology such as electronic, optical, biological, computational and mathematical etc. such that the intrinsic fault tolerance of the model is retained. As well, extra techniques specific to the fabrication technology can be practical apply to further improve the reliability result of the system.

In large systems, due to the large quantity of individual units, the defeat of a few units is unlikely to cause any noticeable reduce in accuracy in the overall system. Indeed, some work has shown that unit losses of up to 40% can be tolerated in the Hopfield model [10]. This tends to propose that non-critical mechanism within the neural network system need not be particularly reliable due to the inherent in generally fault tolerance also that is noted by Belfore and Johnson [11]. However, makes a contentious claim that because of the large number of components "neural computers will need to be designed with high quality components" This seems most unlikely [12]. When neural networks are included in commercial

systems, it will be necessary that the reliability of the implementation can be assessed. So, although it is vital to analyze neural network models at the abstract level initially, eventually it will also be very important to analyze various implementation architectures, as well as taking into consideration the elected technology.

6. CONCLUSION:

Artificial Neural Networks are collected of a large number of simple computational units operating in parallel way they have the potential to provide fault tolerance. The conventional approach to achieve fault tolerance has been to duplicate several times the system performance an important job. The fault tolerance matter of feed forward neural networks as a nonlinear constrained optimization problem by defining the concept of a maximally fault tolerant neural network system.

7. REFERENCES:

1. Von Neumann, J. (1956). Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components, in C. E. Shannon & J. McCarthy (Eds.), Automata Studies, pp. 43-98, Princeton, USA: Princeton University Press.
2. Green, A. E. & Bourne, A. J. (1972). Reliability Technology, New York, USA: John Wiley & Sons.
3. Sequin, C. and Clay, D., "Fault-Tolerance in Artificial Neural Networks", Proc. IJCNN 90, San Diego 1, pp.703-708 (June 1990).
4. Neti, C., Schneider, M.H. and Young, E.D., "Maximally fault-tolerant neural networks and nonlinear programming", Proceedings of IJCNN-90, San Diego **2**, pp.483-496 (June 1990).
5. Segee, B.E. and Carter, M.J., "Comparitive Fault Tolerance of Parallel Distributed Processing Networks (Debunking the Myth of Inherent Fault Tolerance)", Intelligent Structures Group Report ECE.IS.92.07 (1992).
6. Clay, R.D. and Sequin, C.H., "Limiting Fault-Induced Output Errors in ANN's", IJCNN-91, Seattle, supplementary poster session (1991).
7. Bugmann, G., Sojka, P., Reiss, M., Plumbley, M. and Taylor, J.G., "Direct Approaches to Improving the Robustness of Multilayer Neural Networks", Proceedings of the International Conference on Artificial Neural Networks, Brighton UK (1992).
8. Prater, J.S. and Morley Jr., R.E., "Improving Fault Tolerance in Feed forward Neural Networks", submitted to IEEE Transactions on Neural Networks, in review.
9. Heng-Ming, T., "Fault Tolerance in Neural Networks", WNN-AIND-90, pp.59 (Feb 1990).
10. Belfore, L.A. and Johnson, B.W., "The fault-tolerance of neural networks", The International Journal of Neural Networks Research and Applications **1**, pp.24-41 (Jan 1989).
11. Moore, W.R., "Conventional Fault-Tolerance and Neural Computers" pp. 29-37 in Neural Computers, ed. C von der Malsburg, Berlin: Springer-Verlag (1988).

