Missouri University of Science and Technology
Scholars' Mine

Mechanical and Aerospace Engineering Faculty Research & Creative Works

Mechanical and Aerospace Engineering

01 Jan 1992

Use of Time Varying Dynamics in Neural Network to Solve Multi-Target Classification

S. N. Balakrishnan
Missouri University of Science and Technology, bala@mst.edu

J. Rainwater

Follow this and additional works at: https://scholarsmine.mst.edu/mec_aereng_facwork

 Part of the Aerospace Engineering Commons, and the Mechanical Engineering Commons

Recommended Citation

S. N. Balakrishnan and J. Rainwater, "Use of Time Varying Dynamics in Neural Network to Solve Multi-Target Classification," *Proceedings of the IEEE 1992 National Aerospace and Electronics Conference, 1992*, Institute of Electrical and Electronics Engineers (IEEE), Jan 1992.
The definitive version is available at https://doi.org/10.1109/NAECON.1992.220538

This Article - Conference proceedings is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Mechanical and Aerospace Engineering Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

USE OF TIME VARYING DYNAMICS IN NEURAL NETWORK TO SOLVE MULTI-TARGET CLASSIFICATION

S.N. Balakrishnan and Jeffrey Rainwater
University of Missouri-Rolla
Rolla, MO

ABSTRACT

Several types of solutions exist for multiple target tracking: multiple hypothesis testing, probability data association techniques, clustering techniques, etc. However these techniques are computation-intensive and in some cases very difficult to operate on-line. In the current study, a back-propagation neural network has been successfully used to identify multiple moving targets using kinematic data (time, range, range-rate and azimuth angle) from sensors to train the network. Preliminary results from simulated scenarios show that neural networks are capable of learning target identification for three targets during the time-period used during training and a time period shortly after.

INTRODUCTION

Currently, there are several methods[1-3] that can be used for the classification and tracking of multiple target systems -- multiple hypothesis testing, probability data association techniques, clustering techniques, etc. Figure (1) shows the data flow of a clustering technique from earlier published work of the first author[1]. It is clear that this process is computation-intensive since with the arrival of new data the entire process must be repeated.

This paper investigates the use of a neural network using a back-propagation algorithm for multiple target tracking[4-5]. Back-propagation networks have been shown to be very effective in pattern classification problems such as character recognition and fault-diagnosis. These problems deal with static systems.

Unlike these cases, the viability of using a back-propagation network for classification in dynamic systems using kinematic data obtained from sensors is investigated in this study.

The goal of the classification project is to correctly identify a target from a set of data for a specified number of targets using current sensor data (time, range, range-rate, and azimuth angle). The network is trained to classify the different targets using previously classified trajectory data. Several different network structures and training parameters are compared in order to find an optimal network.

NETWORK STRUCTURE AND TRAINING

The network being used is a conventional feed-forward network trained by the method of back-propagation. The input layer consists of four processing elements [PE's] each corresponding to one of the trajectory inputs -- time, range, range-rate, and azimuth angle.

The network has a single hidden layer made up of several PE's -- the number varying from 5 to 50. The output layer is made up of one PE for each target to be classified -- in most cases three. The network is trained so that, ideally, each output node's response will be unity for the object it is trained for but zero for all other objects present in the training data. For example, with a three output network (corresponding to the classification of three objects), if object 1 data is processed, the ideal output for the network will be 1, 0, 0 for outputs 1, 2, and 3 respectively. Or if object 2 data is processed, the ideal output will be 0, 1, 0. The error in classification can be measured by the variance from these ideal outputs.

The network simulation program for this project has several parameters which can be used to affect the network training. These parameters include the learning rate, a momentum term and two iteration parameters -- the number of iterations to perform on each set of inputs in the data file and the number of iterations to perform on the entire data file.

The data file consists of sixty points -- twenty points for three targets. The Range-Rate data field is shown in Figure (1). Each column of the data (one for each kinematic input) was normalized to the range -1 to 1 by dividing the entire column by the absolute value of the

CH3158-3/92/0000-0414 $1.00 © 1992 IEEE

largest occurrence in that column. The data file is ordered so that the observations for the targets at a specific time are together.

The network is trained with the first ten points for each target being tracked. The network is trained with a particular observation of an object a specific number of times before moving to the next observation in the data file. The number of iterations chosen for each observation is fixed at twenty, the minimum needed for good network response. The entire data set is trained with an outer loop consisting of 5000 to 10000 iterations.

NUMERICAL ANALYSIS

At the end of the training phase, a production run is made with a set of twenty data points for each object -- the first ten being the trained data and the last ten being flight data taken after the training data in time.

Data produced during the network training phase showed that the learning curve appeared to be somewhat discontinuous or stair-stepped. Figure (9), the learning curve for the reference case, shows the stair-stepped learning process. This pattern has been previously shown in much simpler systems[6]. One interesting feature is that Output 1 has only 1 drop where Outputs 2 and 3 have two drops. These sudden decreases in error can allow efficient training of networks since training algorithms could easily detect such a drop and cease to train the network shortly after the drop to minimize training time. There is one difficulty, however. The different outputs may have different or more than one step before reaching the minimum.

The decreased training iterations does not seem to significantly affect the performance of the network in untrained data. Two networks, one with momentum and one without, were trained with 6000 iterations instead of 10000. There is very little difference between the 6000 iteration trained networks' performance and their counterparts trained with 10000 iterations.

The first parameter to be investigated is the number of PE's in the hidden layer. Runs were made with 5,7,15,20, and 50 PE's in the hidden layer. The fifteen PE structure was found to be the best and so a 4 input node, 15 hidden node and 3 output node network structure trained with a 0.5 learning rate for 10,000 iterations became the reference case. The response for each output is shown in Figures (2), (3) and (4).

There are a few notable trends in the response of the network. First, the network performs almost flawlessly in all three outputs for data in the training time-period. As untrained observations that are farther away from the trained data are processed, the observations are not classified as well and the performance of the network degrades. Two types of degradations are seen to occur. In the first, the output PE continues to classify "its" object correctly but starts to increase its response to other objects as well. This degradation can be seen in Figure (4). In Figure (4), output 3 of the network correctly identifies the object 3 observations but begins to increasingly respond to object 1 observations in object 3's output node.

The second type of degradation is shown in Figure (3). Output 2 starts to decrease its response to object 2's observations. The second output node (corresponding to object 2) will continuously decrease its response to object 2 observations.

Both degradations start to occur almost immediately after all training data has been completed in the production phase (training data was observations from time index 0.2 to 0.245 with non-trained observations after 0.245.) All tests made, no matter what network structure or training parameters, have the same degradation patterns in varying degrees.

The effect of the momentum term on the performance is also examined. A momentum term of 0.8 accelerates the learning process considerably. This can be seen in figure (5). Figure (5) compares three sets of weight values --the reference case after complete training (10000 outer iterations), the case of an identical structured network using a 0.8 momentum term after complete training, and the same momentum case after only 3000 outer iterations of training.

Although the momentum term accelerates learning, it alters the system performance. This can be seen in figures (6), (7) and (8) -- the response of the momentum trained network. Compared with the response of the reference network (figures (2) - (4)), the degradation is greater in two of the three output node responses, and thereby, the useful classification period is limited even more than the reference condition.

The factor that affects the performance the most is the presence of the time input into the network. A network was trained with three input values, leaving the time index out. The response of this network is shown in Figures (10), (11) and (12). Not only is response degradation much larger than the reference case, but abnormalities exist in the trained area also. The network cannot adequately classify the moving targets with only space data (position and velocity). Time as an indexing parameter is very effective.

There is one theory on the cause of the degradations. In reference [7], the author describes how a back-propagation network splits the nth dimensional space (corresponding to n input fields) into hyper-planes. The combination of these hyper-planes produces distinct classification regions for the different objects being classified. Figure (N), (M), and (O) shows the response each output node in the reference network over the entire normalized range of the Range and Range-Rate fields. Figures (N-O) shows these distinct classification regions with very definite boundaries.

Misclassification occurs when an object's data point is outside the proper classification region. Misclassification occurs in static systems because the samples used in training may not encompass the entire region for that object or data points of two objects may be very close together. However, in static systems, there is no danger that as a whole an object's data will move outside these fixed regions.

This is not the case for dynamic systems. In dynamic systems, an object's data points continue to move through the n-dimensional space. However, the classification regions are fixed in the n-dimensional space at the end of the training-phase. This problem can be intensified by object data paths crossing such as in figure (2) showing the range-rate data field verses time.

To have a truly useful target tracking and classification system, the system must be able to classify an unknown number of targets, knowing when an object leaves the theater or when a new object appears. Multiple output networks are at a disadvantage because in order to add another output, the old network must be abandoned and a new network must be trained.

One solution might be to have a series of single output networks instead of one multiple-output network. This would allow a new network to be started for a new object without destroying the training of the other networks. To this end, three single-output networks were trained with the same data as was used in the previously discussed networks. Each network was trained to respond to only one object each.

The performance of the three networks, shown in Figures (13), (14), and (15) was very similar to the reference case. Networks 1 and 2 corresponding to the first two outputs of the reference network performed slightly less than the reference network. The third network, however, did much better. It had a flawless performance for all data tested.

Although the current networks cannot perfectly classify all observations in untrained areas, a satisfactory classification tracking system could be implemented using the back-propagation networks and a "smart" logic system. The system could look at the outputs of the different networks and make certain judgement values.

For instance, at a time instance at t = 0.29, if the system looked at the Network 1 response (Figure 13) it would see a conflict between Object 1 and Object 3 data so it would table Object 1 classification. Next, it would examine Object 2 and seeing no clear classification, would table Object 2 classification also. Looking at network 3's output however it could definitely say the object 3 observation is classified as Object 3. It could then return to Network 1 and resolve the conflict by removing Object 3 data from the possibilities. Since Object 1 data has a higher response than Object 2 data, the object 1 observation could be categorized as Object 1. That would leave object 2 data to be classified as Object 2. Using this type of logic all three objects could be categorized correctly.

FUTURE STUDY

As mentioned earlier, the introduction of new objects into the tracking presents a unique challenge to the use of back-propagation networks in classification of dynamic systems. Future study will focus on the response of networks to previously unknown object observations. Work will also be done on expanding the multiple network idea to classifying an unknown number of objects.

The presence of the response degradations in networks of different size and training methods points to certain inherent deficiencies in the conventional back-propagation network for classifying dynamic systems. Future investigation will focus on the use of less-traditional network structures such as radial-basis function networks[7].

CONCLUSIONS

For dynamic systems, a conventional back-propagation network can correctly classify objects for short periods of time after training with very few trained data points.

This effective classification period can be extended by the use of networks in coordination with "smart" logic systems. Less traditional networks, such as radial-basis networks or an adapted dynamic back-propagation network show even more promise.

The systems not only have potential in systems with a fixed number of objects, but in systems with an indeterminate number of objects.

These and other improvements are being investigated in order to improve the possibilities for neural network classification of dynamic systems.

REFERENCES

[1]Balakrishnan, S.N. and Tapley, B.D., "Multitarget Classification and Estimation Using Clustering Techniques," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 13, no 1, Jan-Feb. 1990.

[2]Chang, C.B., and Tabaczynski, J.A., "Applications of State Estimation to Target Tracking," *IEEE Transactions on Automatic Control*, Vol AC-29, no. 2, Feb 1984.

[3]Fortman, T.E., Barshalom, Y. and Scheffe, "Sonar Tracking of Multiple Targets Using Joint Probability Data Association," *IEEE Journal of Ocean Engineering*, Vol 8, no 3, July 1983.

[4]Rumelhart, D.E. and McClelland, J.L. Eds., *Parallel Distributed Processing*, Cambridge, MA, MIT Press, 1986.

[6]Hush, Don R., Salas, John M., and Horne, Bill, "Error Surfaces for Multi-layer Perceptrons", International Joint Conference on Neural Networks, San Diego California, June 17-21, 1991.

[7]Leonard, James A. and Kramer, Mark A., "Radial Basis Function Networks for Classifying Process Faults," *IEEE Control Systems Magazine*, Vol. 11, no. 4, April 1991.

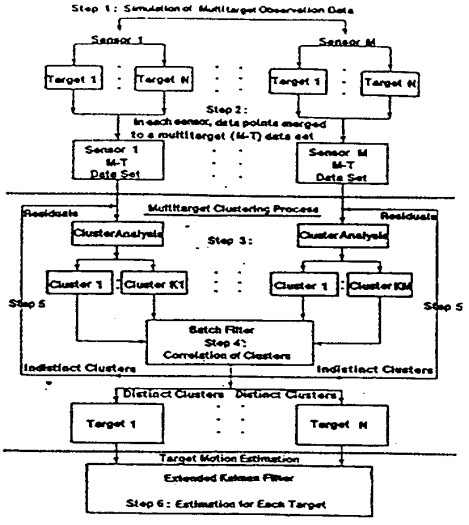

Figure 1. Data Flow.

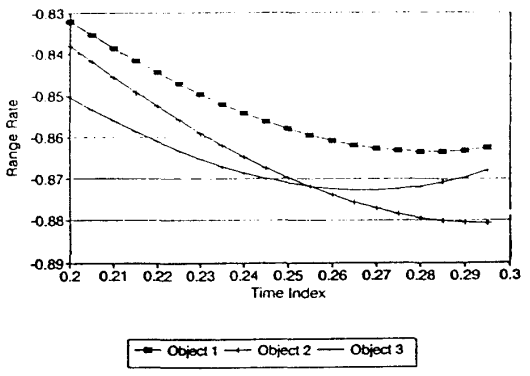

Figure 2. Network Input: Range-Rate vs. Time

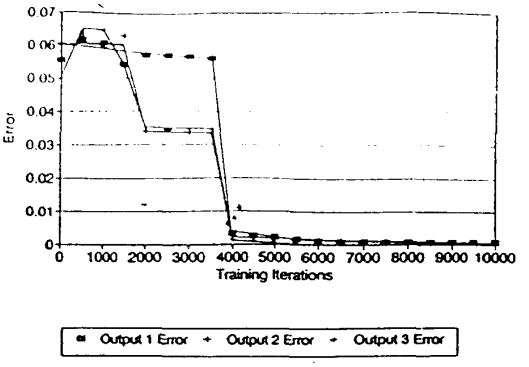

Figure 3. Reference Network Learning Curve

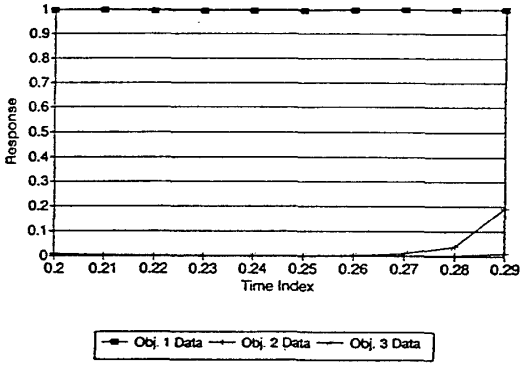

Figure 4. Reference Network Output Node 1 Response

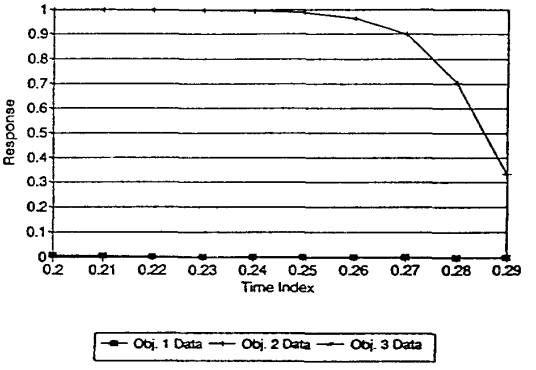

Figure 5. Reference Network Output Node 2 Response

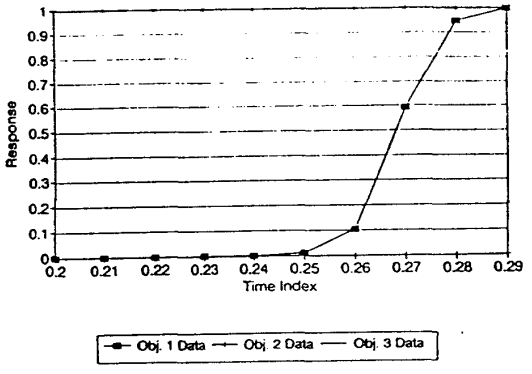

Figure 6. Reference Network Output Node 3 Response

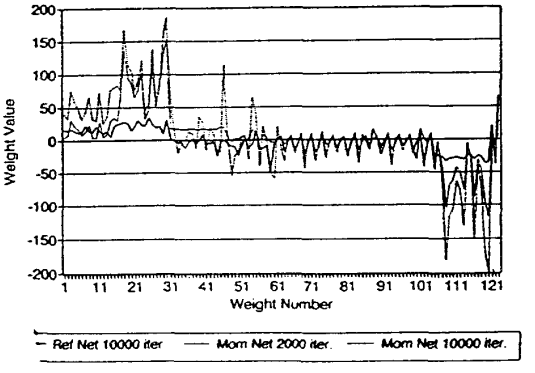

Figure 7. Reference Network & Momentum Network Weights

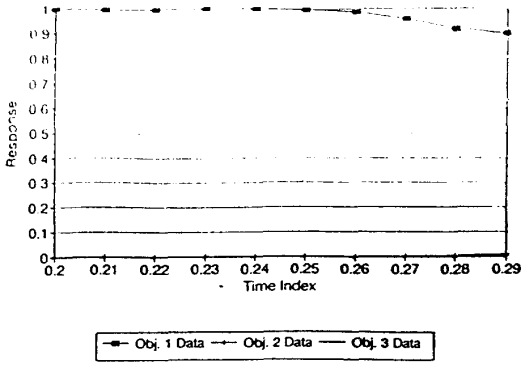

Figure 8. Momentum Trained Network Output Node 1 Response

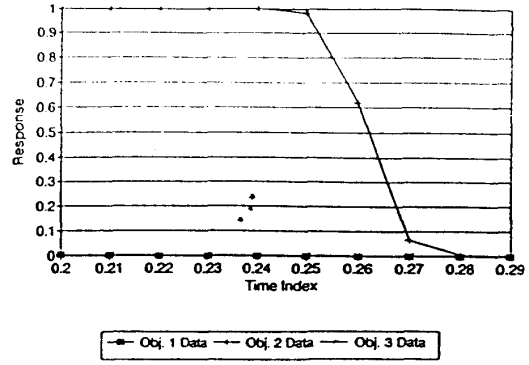

Figure 9. Momentum Trained Network Output Node 2 Response

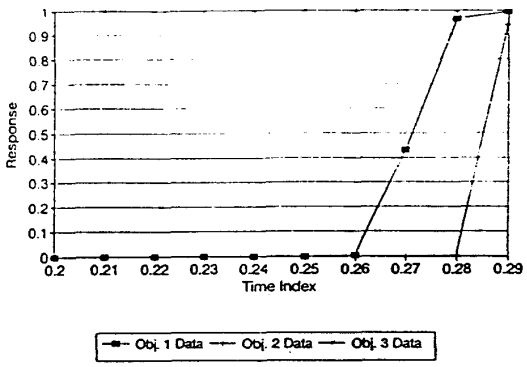

Figure 10. Momentum Trained Network Output Node 3 Response

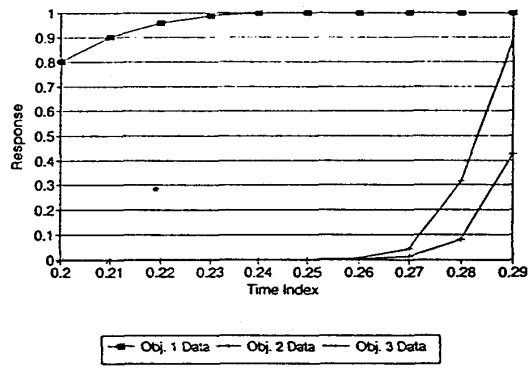

Figure 11. Network Trained w/o Time Output Node 1 Response

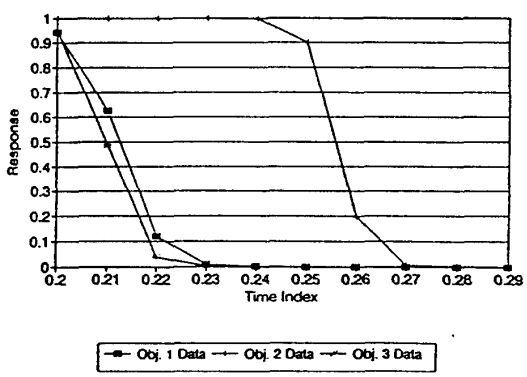

Figure 12. Network Trained w/o Time Output Node 2 Response

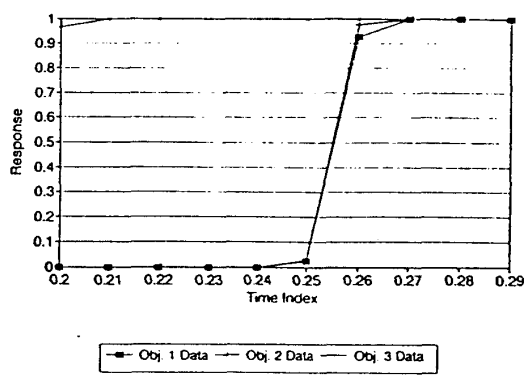

Figure 13. Network Trained w/o Time Output Node 3 Response

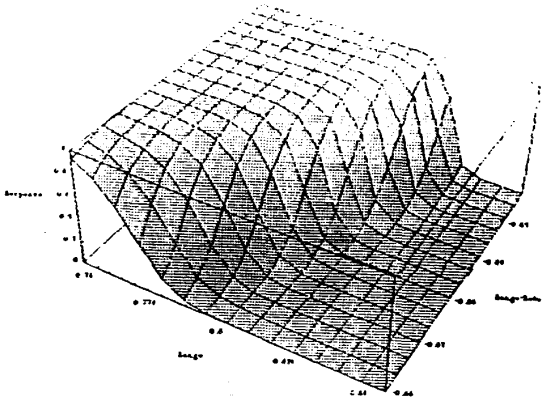

Figure 14. Ref. Network
Output Node 1 Response Surface

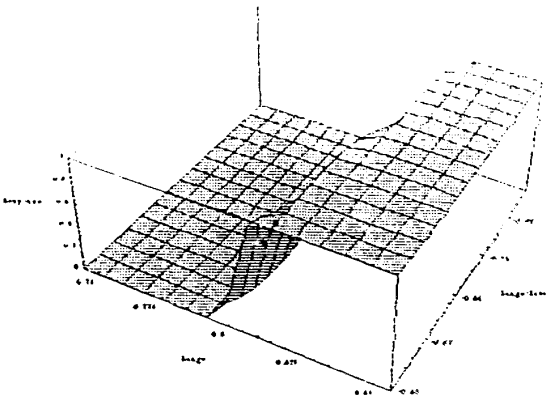

Figure 15. Ref. Network
Output Node 2 Response Surface

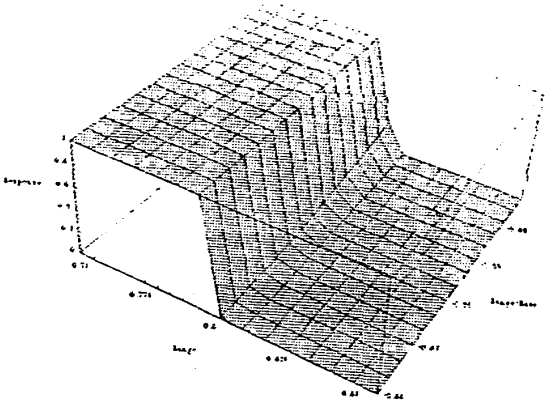

Figure 16. Ref. Network
Output Node 3 Response Surface

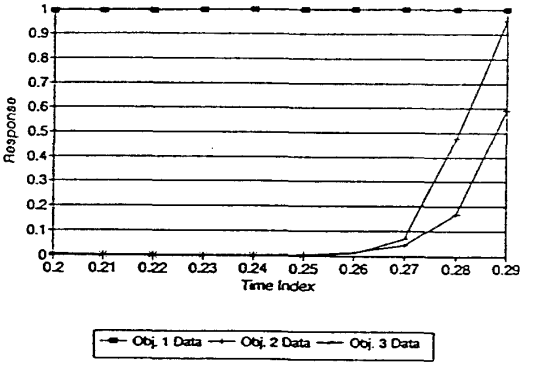

Figure 17. Single Output Net.
Response - Object 1 Trained

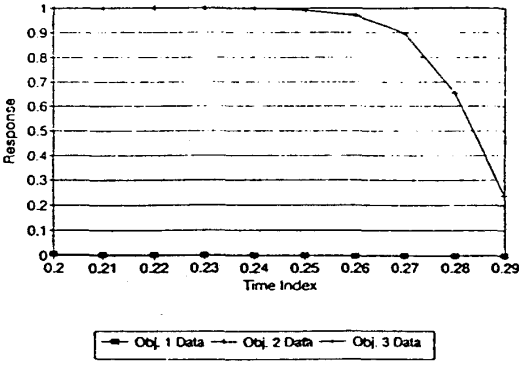

Figure 18. Single Output Net.
Response - Object 2 Trained

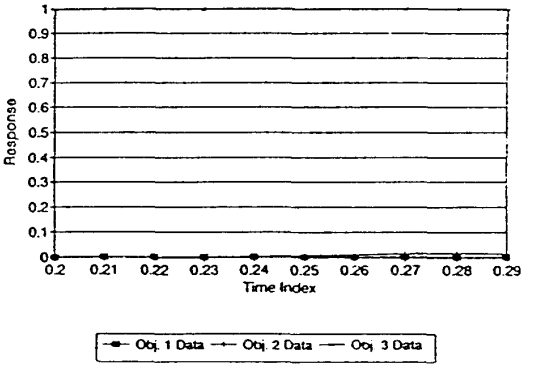

Figure 19. Single Output Net.
Response - Object 3 Trained