A Fast Adaptive Neural Network System for Intelligent Control

Anthony Zaknich, Member IEEE, Associate Member AES
Yianni Attikiouzel, Fellow IEEE, Fellow IEE, Fellow IEAust

**Centre for Intelligent Information Processing Systems (CIIPS),
Department of Electrical and Electronic Engineering,
The University of Western Australia, Nedlands 6907, Western Australia.
Email: tonko@ee.uwa.edu.au**

ABSTRACT

An intelligent control system needs to adapt to new dynamics very quickly but also retain knowledge of past dynamics to be able to act effectively and quickly for repeat occurrences. One solution is to model the system with two neural networks in parallel whereby one network is trained a priori with a wide range of historical dynamics while the second one, is allowed to adapt itself to make up the differences between the first model and the real-time dynamics. Within this scheme, as the second network is called to adapt itself, the first one can be progressively trained to learn the new dynamics without adversely affecting the old training. A strategy of this type can be achieved very effectively using the Modified Probabilistic Neural Network because it is constructed with local radial kernel functions and its adaptation mechanism is computationally simple and very fast. This is demonstrated using a complex nonlinear system whose characteristics suddenly change after initial training and then switch back to the original characteristics. Comparisons are made with other networks to show the important advantages of the Modified Probabilistic Neural Network.

1.0 INTRODUCTION

An adaptive network can be used to model either a linear system whose parameters are unknown (or changing with time) or a nonlinear system whose model is unknown (or also changing with time). A linear adaptive system will eventually converge to a linear solution over sufficient time and range of input signals. It will continue to adapt, only if the system or noise statistics change. For a nonlinear system, a linear adaptive system can only adapt to a linear approximation at the current operating point. It is possible however, to keep a historical record of the set of linear models for each small region around a set of operating points and then apply an appropriate model as the set point changes. This is called schedule or switching control with multiple models. A nonlinear adaptive network will adapt to a more accurate model at the current operating point, but like the linear adaptive network it cannot generalise this to new operating points, unless a historical record is kept. To ensure a more robust control of nonlinear systems it is desirable to have some historical information about the system over the expected range of operating points in parallel with a fast adaptive network to make up any differences.

In references [1,2] it is shown how a Neural Network (NN) structure as depicted in Figure 1 can be used to model a plant's nonlinear dynamic behaviour when certain information is known a priori and the remaining information needs to be learned. Neural Network 1 is trained to learn the plant's known dynamic behaviour whereas Neural Network 2 is used to learn on-line the initially unknown behaviour. Of course once the new behaviour occurs it would be sensible to accumulate the data and include it as history in the a priori model represented by Network 1. If the system is not changing continually into new operating modes then eventually Network 2 will become superfluous and operation can be effected through Network 1.

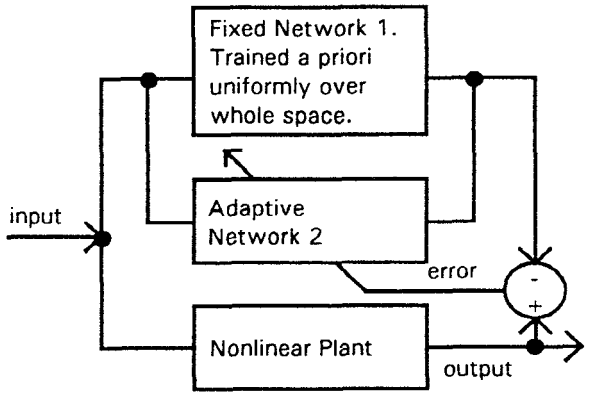

Figure 1. Nonlinear Forward Modelling Scheme

A single Modified Probabilistic Neural Network (MPNN) [3,4] can be used to replace the two neural networks in Figure 1 and still achieve similar functionality. This can be done because the MPNN is based on a set of radial basis functions which provide the property of localised influence. This allows the learning system to develop and refine its control very

0-7803-4053-1/97/$10.00 © 1997 IEEE

quickly in one region of the measurement space without affecting its learning in distant regions. The MPNN is initially trained with all available a priori data and then it systematically integrates all new data as it begins to operate. This is demonstrated with an illustrative example of modelling a complex recursive nonlinear plant. The MPNN is compared with a Multi-Layer Perceptron (MLP) to show the practical benefits, including faster adaptation, better retention of past learning, and better ability to deal with variable system noise. The MPNN architecture and adaptation schemes are reviewed in the next two sections followed by the illustrative example and conclusions.

2.0 REVIEW OF THE MODIFIED PROBABILISTIC NEURAL NETWORK

The Modified Probabilistic Neural Network (MPNN) was initially introduced by Zaknich et al in 1991 [3]. It is closely related to Specht's General Regression Neural Network (GRNN) [5] and both are related to Specht's Probabilistic Neural Network (PNN) classifier. The method of the basic MPNN/GRNN has similarities with the method of Moody and Darken [6]; the method of radial basis functions [7]; and a number of other nonparametric kernel based regression techniques stemming from the work of Nadaraya [8] and Watson [9]. If it can be assumed that for each local region in the input space, represented by a centre vector $\mathbf{c}_i$, there is a corresponding scalar output y_i that it maps into, then the general model to use for all forms of the MPNN and even the GRNN is equation (1).

$$\hat{y}(\mathbf{x}) = \frac{\sum_{i=1}^{M} Z_i y_i f_i(\mathbf{x})}{\sum_{i=1}^{M} Z_i f_i(\mathbf{x})} \qquad (1)$$

$\mathbf{c}_i$ centre vector for class i in the input space.
σ learning parameter chosen during training.
y_i output related to $\mathbf{c}_i$ (real valued or quantised).
M number of unique centres $\mathbf{c}_i$.
Z_i number of vectors $\mathbf{x}_j$ associated with centre $\mathbf{c}_i$.
NS total number of training vectors (Sum of Z_i).
$f_i(x)$ is a radial basis function.

A Gaussian radial basis function is often used for $f_i(x)$ as defined in equation (2) but there are many other suitable radial basis functions which can be chosen in place of the Gaussian function. All the radial basis functions have exactly the same learning parameter σ chosen during training.

$$f_i(\mathbf{x}) = \exp\frac{-(\mathbf{x}-\mathbf{c}_i)^T(\mathbf{x}-\mathbf{c}_i)}{2\sigma^2} \qquad (2)$$

Equation (1) represents the GRNN if all the Z_i=1, the y_i are real valued, the centre vectors $\mathbf{c}_i$ are replaced with individual training vectors $\mathbf{x}_i$ and M=NS. The MPNN can be seen as a kind of size reduced GRNN as it has virtually the same performance specifications as the GRNN, but in a more computationally efficient structure. There are many ways to select of the parameters M, Z_i and $\mathbf{c}_i$ for the MPNN but in all cases the selection is done very simply with minimal computational requirements [4,10].

3.0 MPNN/GRNN ADAPTATION SCHEME

The basic adaptive structure for the MPNN/GRNN is depicted in Figure 2. The actual input and desired data are fed directly into the MPNN/GRNN storage buffers to be used in the network architecture. In the case of the GRNN the data $\{\mathbf{x}_i \rightarrow y_i \mid i=1,...NS\}$ are used without any modification, but in the case of MPNN they are reduced with minimal computational complexity [4,10] to $\{\mathbf{c}_i \rightarrow y_i \mid i=1,...M\}$ before storage. Adaptation involves finding the optimal σ giving the minimum mse for some fixed number NUM of known sample vectors passing through the network. For most problems the (mse vs σ) curve is smooth with a single minimum and it easy to find a suitable σ.

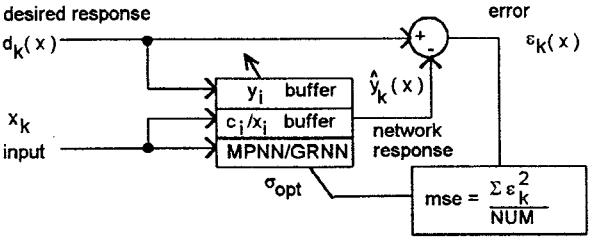

Figure 2. Basic MPNN/GRNN Adaptive Structure

In the case of the GRNN the training data simply flows through a fixed sized delay line buffer so that old data are lost as new data come in. This ensures fast system adaptation to changing statistics because the network is constructed from data of the recent past only. In the case of the MPNN for a new data pair $\{\mathbf{x} \rightarrow y_i\}$, categorised as belonging to the centre i, it adds to the training data according to equations (3) and (4)

$$Z_i^{new} = Z_i^{old} + 1 \qquad (3)$$

only if new training vector $\mathbf{x}$ belongs to $\mathbf{c}_i$.

$$\mathbf{c}_i^{new} = \frac{(Z_i - 1)\,\mathbf{c}_i^{old} + \mathbf{x}}{Z_i} \qquad (4)$$

if $Z_i > 0$ and new training vector $\mathbf{x}$ belongs to $\mathbf{c}_i$, else if $Z_i = 0$, $\mathbf{c}_i^{new} = \mathbf{x}$, the associated y_i is also added.

The new vector pair $\{\mathbf{x} \rightarrow y_i\}$ is deemed to belong to the class i if the quantised characterisation vector made from the quantised elements of $\{\mathbf{x} \rightarrow y_i\}$ matches the specific characterisation vector defining class i. The exact nature of that quantised characterisation vector depends on the characterisation Method A or B [10] chosen. For Method A it only depends on the quantised output values whereas in Method B it includes both the input and output quantised vector elements.

Equations (3) and (4) are strictly only valid for stationary data statistics. Eventually, the centre vectors, $\mathbf{c}_i^{new}$, will converge at which time the accumulation may be stopped or continued with no detrimental effects. One way to solve this problem when Method B is used is to also check whether $\mathbf{x}$ belongs to the quantised input part of the characterisation vector. If it does, but the output part does not match, then reduce the corresponding Z_i by 1; if $Z_i \geq 1$ create a new class. In this way old irrelevant training will eventually be extinguished and replaced with new learning without affecting any other learning in the network.
For changing statistics or for self regulation it is also possible to introduce a forgetting factor τ expressed in terms of a discrete number of update sample points into equations (3) and (4) as follows.

$$Z_i^{new} = \frac{(\tau - 1)\, Z_i^{old} + 1}{\tau} \qquad (5)$$

if new training vector $\mathbf{x}$ belongs to $\mathbf{c}_i$.

$$Z_i^{new} = \frac{(\tau - 1)\, Z_i^{old}}{\tau} \qquad (6)$$

if new training vector $\mathbf{x}$ does not belong to $\mathbf{c}_i$.

$$\mathbf{c}_i^{new} = \frac{(Z_i - 1)\, \mathbf{c}_i^{old} + \mathbf{x}}{Z_i} \qquad (7)$$

if $Z_i > 0$ and new training vector $\mathbf{x}$ belongs to $\mathbf{c}_i$, else if $Z_i = 0$, $\mathbf{c}_i^{new} = \mathbf{x}$, the associated y_i is also added.

While the buffers are being loaded as described above, σ is adapted to maintain optimal performance. For stationary signal and noise statistics once the buffers are filled for the GRNN, or the centres have converged for the MPNN, and an optimum σ_{opt} has been established, it can then be fixed along with the buffer data. Otherwise, σ must be periodically adapted as the data flows through the buffers.

4.0 AN ILLUSTRATIVE EXAMPLE

Equation (8) (Narendra and Parthasarathy's [11] type 4 model) was used to model the complex nonlinear system as described below, where $x(n)$ is the input and $y(n)$ is the output sequence.

$$y(n+1) = \frac{y(n)y(n-1)y(n-2)x(n-1)[y(n-2)-1] + x(n)}{1 + y^2(n-1) + y^2(n-2)} \qquad (8)$$

Initially the system output for state 1 is simply $y=y(n+1)$. At some later time it suddenly changes to state 2: $y=(y(n+1))^2$ for $y(n+1)\geq 0$ and, $y=-(y(n+1))^2$ for $y(n+1)<0$ and then later in time switches back to the initial state 1. For each state the same inputs produce distinctly different outputs which provides an extreme case study.

Training and testing digital signal sequences of 500 points each were simulated according to equations (10) and (11) respectively.

$$x(n) = \sin(\frac{2\pi k}{250}) \qquad (10)$$

$$x(n) = 0.8\sin(\frac{2\pi k}{250}) + 0.2\sin(\frac{2\pi k}{25}) \qquad (11)$$

In addition to these signals a further 500 points of a random sequence bandlimited to 0-4.0 Hz (assuming a Nyquist sampling frequency of 100 Hz) for training and another 500 independent points for testing, for each plant, were also simulated. These sequences along with their desired outputs are shown in Figures 3 and 4. There were 1000 training and 1000 testing vector pairs for each of the two states.

The neural network input vector $\mathbf{x}$ was constructed according to equation (12) indicating a recursive design.

$$\mathbf{x} = (x(n), x(n-1), y(n), y(n-1), y(n-2))^T \qquad (12)$$

Given the training and testing vectors for state 1 a MLP (5-20-1) with 20 hidden nodes, a GRNN with 1000 hidden nodes, a Method B MPNN with 662 hidden nodes and input quantisation level N_X=2 and output quantisation level N=4096 and a Method B MPNN with 149 hidden nodes and input quantisation level N_X=2 and output quantisation level N=128 were trained a priori and the results are shown in Table I. The table shows the number of training iterations, the mean squared error (mse) between the outputs and desired outputs, the training and vector evaluation times for each network implemented in C and run on an IBM compatible PC Pentium 90. Table I also shows testing results when random noise with a uniform distribution and variance of 0.021 is added to the testing input vectors.

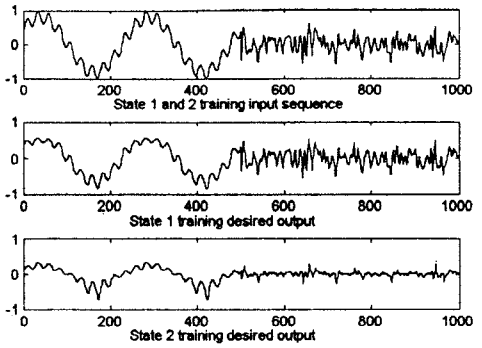

Figure 3. Training sequences

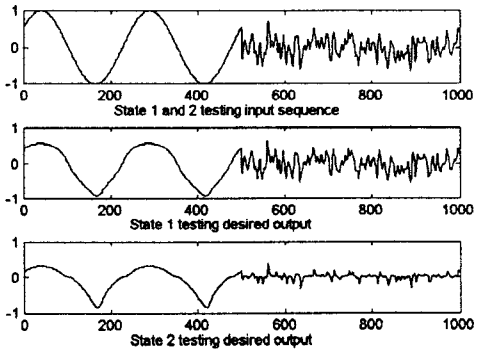

Figure 4. Testing sequences

If the parallel setup of Figure 1 is adopted with network 1 trained a priori for state 1 Network 2 adaptively learns the difference between state 1 and state 2 as state 2 training data begins to occur. Then network unlearns it when state 1 data is reintroduced. The training times shown in Tables I and II are typical of the times required for Network 2 to adapt itself either way and they represent the extreme case of adapting to distinctly different dynamics. The MPNN/GRNN methods adapt to their optimal points after exactly 1000 new points which represents the total number of training points for each state. The MLP on the other hand takes considerably longer in terms of both training time and number of training iterations. The MLP used simple gradient descent learning (gain factor of 0.01 and momentum factor of 0.001) which is the slowest learning method. However, whatever method is adopted it will never learn faster than the MPNN/GRNN which simply takes the new training data a point at a time and systematically adds it to its structure according to adaptation equations (3) and (4) or equations (5), (6) and (7).

Instead of using the parallel setup of 1 a single MPNN can be used in conjunction with adaptation equations (3) and (4) plus the extra processing suggested for Method B. This would achieve exactly the same results as shown in Tables I and II. This approach offers considerable advantage for intelligent control applications. New dynamics are integrated and retained in the structure and are not extinguished except in the specific case when they should be replaced to avoid erroneous operation. Any changes only affect local regions in the operating space leaving historical data in other regions in tact. With only very minor modifications it is also possible to build a system similar to schedule or switching control with multiple models by identifying separate regions of the MPNN space which have been trained with specific dynamics. An identification vector according to equation (13) can identify the different dynamic states and activate the respective region of the MPNN space.

$$\mathbf{x} = (x(n), x(n-1), y(n), y(n-1), y(n-2), y)^{\mathrm{T}} \qquad (13)$$

TABLE I

Net. Type	Data Set	Train iter./ passes	mse/2	σ	Train time in sec	Vect. eval. in sec
MLP 5-20-1	train test noisy	400,000	0.00012 0.00031 0.00820	- - -	132.4	0.00033
GRNN 5-1000-1	train test noisy	1	- 0.00022 0.00314	- 0.032 0.135	9.72	0.00972
MPNN Meth. B 5-662-1 N_x=2 N=4096	train test noisy	1	- 0.00024 0.00314	- 0.029 0.140	6.32	0.00632
MPNN Meth. B 5-149-1 N_x=2 N=128	train test noisy	1	- 0.00106 0.00357	- 0.034 0.111	1.81	0.00181

Training results for state 2 are shown in Table II.

TABLE II

Net. Type	Data Set	Train iter./ passes	mse/2	σ	Train time in sec	Vect. eval. in sec
MLP 5-20-1	train test noisy	1300000	0.00016 0.00036 0.00450	- - -	430.3	0.00033
GRNN 5-1000-1	train test noisy	1	- 0.00035 0.00153	- 0.023 0.108	9.72	0.00972
MPNN Meth. B 5-548-1 N_x=2 N=4096	train test noisy	1	- 0.00035 0.00153	- 0.024 0.108	4.23	0.00423
MPNN Meth. B 5-117-1 N_x=2 N=128	train test noisy	1	- 0.00051 0.00146	- 0.068 0111	1.10	0.00110

5.0 CONCLUSIONS

It has been shown how the MPNN is very suitable for intelligent control applications. It has extremely fast adaptation ability, the ability to absorb new local learning without adversely affecting previous global learning and very good noise tolerance. The MPNN is a general regression method which can be implemented in a simple parallel hardware structure.

6.0 REFERENCES

[1] Warwick, K., Irwin, G. W. and Hunt, K. J., editors for "Neural networks for control and systems", Peter Perigrinis, London, 1992.

[2] Miller, W. T.., Sutton, R. S. and Werbos, P. J., "Neural networks for control", MIT Press, MA, 1990.

[3] Zaknich, Anthony, deSilva, Christopher and Attikiouzel, Yianni, "The probabilistic neural network for nonlinear time series analysis", IEEE International Joint Conference on Neural Networks (IJCNN), Singapore, 17-21st November 1991, pp. 1530-1535.

[4] Zaknich, Anthony and Attikiouzel, Yianni, "Time series characterisation schemes for the modified probabilistic neural network", Australian Journal of Intelligent Information Processing Systems (AJIIPS), Vol. 2, No. 2, Winter 1995, pp. 1-11.

[5] Specht, D. F., "A general regression neural network", IEEE Transactions on Neural Networks, Vol. 2, No. 6, November 1991, pp. 568-576.

[6] Moody, J. and Darken, C., "Fast learning in networks of locally-tuned processing units", Neural Computation, Vol. 1 No. 2, September 1989, pp. 281-294.

[7] Powell, M. J. D., "Radial basis functions for multivariate interpolation: A review", Technical Report DAMPT 1985/NA12, Department of Applied Mathematics and Theoretical Physics, Cambridge University, England, 1985.

[8] Nadaraya, E. A., "On estimating regression", Theory Probab. Appl., 9, 1964, pp. 141-142.

[9] Watson, G. S., "Smooth regression analysis", Sankhya Ser. A, 26, 1964, pp. 359-372.

[10] Zaknich, Anthony and Attikiouzel, Yianni, "A modified probabilistic neural network signal processor for nonlinear signals", Proceedings of the 13th International Conference on Digital Signal Processing (DSP-97), Santorini, Greece, 2nd-4th July, 1997.

[11] Narendra, K. S.. and Parthasarathy, K., "Identification and control of dynamical systems using neural networks", IEEE Transactions on Neural Networks, Vol. 1, March 1990, pp. 4-27.