A Modified Probabilistic Neural Network (PNN) for Nonlinear Time Series Analysis.

Anthony Zaknich, Member IEEE, Associate Member AES
Christopher J. S. deSilva
Yianni Attikiouzel, Senior Member IEEE, Fellow IEE
Department of Electrical and Electronic Engineering
The University of Western Australia
Nedlands 6009,Western Australia

Abstract
Donald Specht introduced a one-pass learning algorithm called the Probabilistic Neural Network (PNN) for classification, mapping and associative memory. A modified PNN is proposed that can be used for nonlinear time series analysis without loss of the advantages offered by Specht's PNN architecture. It is shown how the Gaussian Radial Basis Function, expressed as a Parzen probability density function (pdf) estimator, can be used to estimate and implement nonlinear mappings applied to time series data. The performance of this modified PNN is demonstrated by showing its effectiveness in smoothing a sinusoidal signal which has been compressed in amplitude and then corrupted with wideband non-gaussian noise. The network is also compared with the multi-pass learning Backpropagation Network (BPN), and relative merits of the proposed modified PNN are discussed.

Introduction
The main purpose of this paper is to show how the Probabilistic Neural Network (PNN) architecture proposed by Specht can be easily adapted for nonlinear time series analysis. This is done by exploiting the links between the PNN architecture and Gaussian Radial Basis Functions. A short summary of the PNN classifier is given, as the modified PNN for time series analysis shares its features. This is followed by a theoretical development which is the basis for the modified network. Experimental results are reported and analysed for a nonlinear noisy time series filtering problem using the modified PNN and a BPN for comparison.

PNN Classifier
Specht's PNN [1,2] classifier is a three-layer, feed-forward, one-pass, learning network that uses sums of Gaussian distributions to estimate the class probability density functions as learned from training vector sets. Consequently, the PNN is able to make a classification decision in accordance with the Bayes strategy for decision rules and to provide probability and reliability measures for each classification. Learning involves choosing a single suitable smoothing factor which is the common standard deviation for all the Gaussians). The PNN uses one of a class of probability density function estimators which asymptotically approach the underlying parent density provided that it is smooth and continuous [3]. The network is tolerant of erroneous training vectors and sparse data samples can be adequate for optimal performance. It is both easy to use and fast for moderately sized data bases. The major disadvantage of the PNN is that all training vectors must be stored and used to classify new vectors, thus requiring large memories for many practical problems. This is not a severe disadvantage if the PNN is implemented in a parallel hardware structure where memory is relatively inexpensive.

Theory: Parzen pdf Estimators and Gaussian Radial Basis Functions
The process of learning a linear or nonlinear input-output mapping from a set of examples can be seen as developing an approximation for a multivariate function. When dealing with pattern classification problems methods such as Parzen, pdf estimators [3] are commonly used, whereas for continuous mappings regularization techniques (which are closely related to the interpolation technique of Radial Basis Functions) can be used [4]. These two methods are closely related to each other as well to some neural networks including the BPN and PNN. The following development shows this relationship and

CH 3065-0/91/0000-1530 $1.00 © IEEE

leads to the modified PNN for nonlinear time series analysis. The equation for the Parzen pdf estimator which can be used in Specht's PNN classifier is:

$$f_i(\mathbf{x}) = (1/((2\Pi)^{p/2}\, s^p\, M_i)) \sum_{j=1}^{M_i} \exp(-(((\mathbf{x}-\mathbf{x}_{ij})^T (\mathbf{x}-\mathbf{x}_{ij}))/(2\, s^2))) \quad (1)$$

where:

T indicates the transpose.
i indicates the class number.
j indicates the pattern number.
$\mathbf{x}_{ij}$ is the j^{th} training vector from class i.
M_i is the number of training vectors in class i.
p is the dimension of vector $\mathbf{x}$.
s is the smoothing factor, standard deviation.
$f_i(\mathbf{x})$ = sum of multivariate Gaussians centred at each of the vectors $\mathbf{x}_{ij}$, for the i^{th} class pdf estimate.

The Bayes decision $d(\mathbf{x})=C_i$ is made (i.e. multi-dimensional vector $\mathbf{x}$ belongs to class i) if $l_i.h_i.f_i(\mathbf{x}) > l_k.h_k.f_k(\mathbf{x})$ for all j not equal to k. The l_i are the losses associated with making an incorrect decision for each class i. In many cases the losses can be considered equal so they may be cancelled from the equation. The h_i are the a priori probabilities of occurrence of class i vectors.

Poggio and Girosi [4] have developed a theoretical framework for a class of three-layer regularization networks which includes the Radial Basis Function methods as a special case. Schoenberg's theorem [5] on positive definite functions shows the radial basis function expansion to be:

$$O(X) = \sum_{i=1}^{NS} c_i.g(||X-X_i||) \quad (2)$$

The unknown coefficients c_i are determined by solving $O(\mathbf{x}_i) = O_i (i=1,...NS)$ where the O_i are the known or desired values of the multivariate function at specific inputs $\mathbf{x}_i$ and NS is the total number of known function values. Michelli [6] has justified the use of a number of basis functions g(r) for practical data interpolation problems. One of these, is the Gaussian function: $g(r) = \exp(-(r/a)^2)$ (3)

If equations (2) and (3) are combined and $2.s^2$ is equated to a^2 this results in equation (4) as follows:

$$O(\mathbf{x}) = \sum_{i=1}^{NS} c_i \exp(-(((\mathbf{x}-\mathbf{x}_i)^T (\mathbf{x}-\mathbf{x}_i))/(2\, s^2))) \quad (4)$$

The function $O(\mathbf{x})$ can be used to find estimates of interpolation values at $\mathbf{x}$ for a partially known function. However, we need to solve a slightly different problem. We do wish to solve for interpolation but in our case the known input vectors $\mathbf{x}_i$ form clusters of points with specific probability density distributions and these map into the desired values O_i with their own a priori probabilities of occurrence. Our function $O(\mathbf{x})$ must in some way use the known samples O_i to form a best fit for not only the known samples but also for all subsequent samples which will be generated by the parent process. To show one way this can be achieved we shall develop equation (4) a little further and eventually draw out of it the required pdf estimator as represented in equation (1). If we let:

$$c_i = O_i / [\sum_{k=1}^{NS} \exp(-(((\mathbf{x}-\mathbf{x}_k)^T (\mathbf{x}-\mathbf{x}_k))/(2\, s^2)))] \quad (5)$$

then, as s approaches zero, $O(\mathbf{x})$ approaches the training values O_i exactly with no interpolation. This occurs because the Gaussian basis functions approach delta functions centred at the O_i values. However, as s increases we find that interpolation occurs at the expense of being true to the original O_i fit. Thus, equations (4) and (5) can be used to learn a nonlinear time series mapping by taking a training set of data $\{\mathbf{x}_i \text{--} > O_i | i=1,...NS\}$ and finding an s value which minimises the mean squared error, (mse) between $O(\mathbf{x}_j)$ and the desired output O_j for all the points of another independent testing set. We have discovered that if the data sets have sufficiently representative samples of the parent process then the function $O(\mathbf{x})$ at the optimal s will perform a satisfactory mapping.

So far we have taken each $\{\mathbf{x}_i \text{-->} O_i\}$ pair as a separate and independent process sample without regard for any clustering or relationship amongst sets of samples. In actual fact, putting a radial basis function at the centre of each sample automatically builds up a multivariate function which is proportional to the pdf estimates of the natural clusters of samples. We can show this for a discretely sampled time series by assuming that for each possible digital quantization level y_i we can build up an i^{th} class pdf estimate in accordance with equation (1). If we now say that the estimate of the multivariate function $O(\mathbf{x})$ is denoted by Y(n) it can be shown that equations (1), (4) and (5) can be merged to form equation (7). This is done by taking M_i groupings of equal O_k values and their associated input vectors $\mathbf{x}_{ij}$ and giving them all the same value $y_i = O_k$ for each class i. Equation (6) in conjunction with the fact that h_i, the a priori of occurrence of y_i, is directly proportional to M_i, the number of vectors in class i, provides the required bridge between equations (1), (4) and (5) and equation (7).

$$\sum_{i=1}^{NS} O_i.\exp(-(((\mathbf{x}-\mathbf{x}_i)^T(\mathbf{x}-\mathbf{x}_i))/(2\,s^2))) = \sum_{i=1}^{N} y_i \cdot \sum_{j=1}^{M_i} \exp(-(((\mathbf{x}-\mathbf{x}_{ij})^T(\mathbf{x}-\mathbf{x}_{ij}))/(2\,s^2))) \quad (6)$$

In this present study all time series samples O_k with equal values and on positive slopes were classed together, while those of the same value but on negative slopes were taken together into a separate class. For simple time series signals, like sinusoids, this kind of grouping is adequate but for other signals where there may be a number of different slopes associated with the same O_k value, a different grouping rule and a slight modification to equation (7) is needed to maintain good performance. This, however, is not the subject of this present work but will be reported in a future publication. It is sufficient to say that the desired groupings must be chosen properly before the modified PNN can perform its function adequately.

$$Y(n) = \left[\sum_{i=1}^{N} y_i\, h_i\, f_i(\mathbf{x}) \right] / \left[\sum_{k=1}^{N} h_k\, f_k(\mathbf{x}) \right] \quad (7)$$

where for our chosen filter experiment:

$\mathbf{x}$= [X(n-5),X(n-4),X(n-3),X(n-2),X(n-1),X(n),X(n+1),X(n+2),X(n+3),X(n+4),X(n+5)], vector.

N is the number of classes.

y_i is the desired output for training vectors in class i.

h_i is the a priori probability of occurrence of class i vectors.

$NS = \sum_{i=1}^{N} M_i$, which is the total number of training samples.

Equation (7) is the basis of our modified PNN shown in Figure 3. Although in the equation, N represents all the possible classes, it is possible to achieve minor improvements in performance by choosing a lower value of N and using only the N highest $f_i(\mathbf{x})$ estimates for each input vector $\mathbf{x}$ in turn. It is possible to look at equation (7) as a weighted sum of all y_i multiplied by the relative probabilities that vector $\mathbf{x}$ belongs to each class i. The estimate Y(n) is weighted more by classes which are closest to $\mathbf{x}$ but since all classes contribute to it this performs an interpolation with a beneficial smoothing effect.

Experimental Results

To test the modified PNN for nonlinear time series analysis a system was built according to the block diagram shown in Figure 1. Inspiration for the test set up was gained from the works of Uncini et al [7] and Hoyt et al [8]. The system was driven by a continuous sinusoidal signal S(t) at a constant frequency of 1.000 KHz. The sinusoid was amplified, then compressed (by an approximate exponential compressor made up of back-to-back signal diodes) Z(t) and finally corrupted with three levels of wideband non-gaussian noise N(t), resulting in signal X(t). Example plots of the three noise levels (a) low, (b) medium and (c) high and the compressed signal Z(n) are shown in Figure 4. All three levels of noise had approximate zero means and skews of -0.36. The three signal to noise ratios represented are (a) 12.2 dB, (b) 3.7 dB and (c) -0.4 dB.The S(t) and X(t) continuous signals were both digitised with an 8 bit analog to digital converter (ADC) at a sample rate of 30.1 KHz. The ADC level of 128 represented a signal voltage of zero volts,with 255 being the most positive voltage and 0 the most negative voltage.

Three independent data sets of digitised sequences of S(t) and X(t), denoted as S(n) and X(n) respectively, were then used to train, test and evaluate both the modified PNN and a four layer BPN. The three data sets are referred to as the Training, Testing and Evaluation sets. Figures 5 and 6 show the results of passing the same Testing signal sequence X(n) through the PNN and BPN filters each trained with the Training data set. Figure 5 shows the output from the PNN filter for N=129. Figure 6 shows the best output of the BPN filter trained over 8 million iterations with a gain factor of 0.1 and momentum factor of 0.01. The Training set was composed of 504 discrete samples and both the Testing and Evaluation sets were composed of 501 samples each. The Evaluation set was used to verify the performances of both filters after training and testing. The neural network filters in both cases were configured as smoothing filters as shown in Figure 2. Each discrete output sample Y(n) maps from an eleven coefficient input vector **x** made up of the five discrete time series samples immediately prior to and following the current discrete sample X(n) of the digitised nonlinear system output. The resulting smoothed signals Y(n), may be compared with each other and the original source signal S(n). The mean squared errors of the source signal S(n) compared with each of the filtered outputs Y(n) for each network filter are given in the tables below. Quoted times are the running times of software implementations written in C and executed on an 80386 AT compatible PC with a clock frequency of 33 MHz. The N referred to in the modified PNN FILTER table indicates the number of highest class pdf estimates used for each filtered vector, where, 129 indicates all the classes formed from the 504 training samples and 1 indicates only the single highest class. Initially all the pdfs are calculated for each input vector **x**, according to equation (7), but then only the highest N are used in the equation.

Modified PNN FILTER (The Training set is used for the PNN weights $\mathbf{x}_{ij}$, Input Layer=11 nodes)

Data Set	MSE	N	Best s	Training Time	Vector Execution Time
Testing	0.000274	129	0.21-0.22	60.70 secs	0.121 secs per vector
Evaluation	0.000153	129	0.16-0.17	60.70 secs	0.121 secs per vector
Testing	0.000273	64	0.21	67.28 secs	0.134 secs per vector
Evaluation	0.000153	64	0.16-0.17	67.23 secs	0.134 secs per vector
Testing	0.000271	32	0.21-0.22	67.01 secs	0.134 secs per vector
Evaluation	0.000153	32	0.16-0.17	67.01 secs	0.134 secs per vector
Testing	0.000277	16	0.22	66.96 secs	0.134 secs per vector
Evaluation	0.000149	16	0.17-0.18	66.96 secs	0.134 secs per vector
Testing	0.000293	8	0.20	66.90 secs	0.133 secs per vector
Evaluation	0.000161	8	0.17	66.85 secs	0.133 secs per vector
Testing	0.000362	4	0.21	66.90 secs	0.134 secs per vector
Evaluation	0.000204	4	0.17	66.85 secs	0.133 secs per vector
Testing	0.000482	2	0.24	66.85 secs	0.133 secs per vector
Evaluation	0.000279	2	0.20	66.79 secs	0.133 secs per vector
Testing	0.000658	1	0.21	60.31 secs	0.120 secs per vector
Evaluation	0.000413	1	0.19	60.37 secs	0.121 secs per vector

BPN FILTER (1^{st} Layer=11 nodes, 2^{nd}=21 nodes, 3^{rd}=11 nodes, Last=1 node)

Data Set	MSE	Iterations	Training Time	Vector Execution Time
Training	0.000169	$2x10^6$	877.71 mins	
Testing	0.000210			0.014 secs per vector
Evaluation	0.000198			0.014 secs per vector
Training	0.000153	$4x10^6$	1755.42 mins	
Testing	0.000198			0.014 secs per vector
Evaluation	0.000192			0.014 secs per vector
Training	0.000154	$8x10^6$	3510.84 mins	
Testing	0.000177			0.014 secs per vector
Evaluation	0.000164			0.014 secs per vector

To compare the generalised performance of the filters another test signal which was of a similar type but different to the training signal having a similar amplitude and with a linear frequency sweep (chirp) from

650-1550 Hz plus a high noise level (S/N = 0 dB) was sampled. Figures 7 and 8 show the results of filtering this test set of 502 samples through the PNN and BPN filters respectively. The modified PNN filter produced a filtered output Y(n) with a mse=0.001632 at s=0.23 while the best BPN filter produced a mse=0.001071. It is interesting to note that although the modified PNN appeared to give a better overall source signal amplitude recovery the BPN filter achieved a lower mse. Another test involved filtering a compressed signal with no noise added. Although neither network filter had been trained with no noise they both achieved excellent source signal recovery. The modified PNN filter achieved a mse=0.000125 for an s=0.11-0.12 and the best BPN filter achieved a mse= 0.000127.

Network Comparisons and Discussion

Our work has shown that it is possible to perform effective nonlinear time series analysis with a modified PNN classifier architecture. All the usual advantages and characteristics of the PNN are evident. The mse variation with changing s showed fairly smooth but distinct minima around the optimum s values. As the number of allowable quantization levels y_i, were gradually reduced the performance of the modified PNN degraded slowly and gracefully. The same occurred when the number of training samples were reduced, showing that the modified PNN was able to generalise the filtering very well. Comparing the results of the modified PNN for N=1 and N= 2 to N=129 shows that interpolation and smoothing is occurring for N>1. For N=1 the network works as a normal PNN classifier, choosing the most likely y_i as the filter output. The experimental results have shown that the modified PNN and BPN filters achieve reasonably similar performance with the BPN giving an overall lower mean squared error. However, for the software implementations of the filters, the modified PNN required approximately three orders of magnitude less training time but only one order of magnitude more signal filtering time for 504 training samples. One of the major advantages of the modified PNN over the BPN is that it can follow changing or nonstationary waveform statistics quickly and simply by either adding to or replacing old training data with new data as it becomes available.

The present study has shown that equation (7) has valuable application to nonlinear time series analysis, however, a more complete investigation needs to be made to establish its full capabilities and limitations. A better understanding of y_i class grouping rules for various types of nonlinear signals needs to be developed. It is suspected that for more complex signals the class grouping may need to be further sub-divided into sub-groups with similar features. In this case it would first be necessary to choose the best sub-group for each new vector **x** and then only use the classes in that sub-grouping for equation (7) rather than including all the possible trained classes.

References:

[1] Specht, D. F., "Probabilistic Neural Networks For Classification, Mapping, or Associative Memory", IEEE Conference on Neural Networks, Vol. I, San Diego, July 1988, pp. 525-532.

[2] Specht, D. F., "Probabilistic Neural Networks and Polynomial Adaline as Complementary Techniques for Classification", IEEE Transactions on Neural Networks, Vol. I, No 1, March 1990, pp. 111-121.

[3] Parzen, E., "On estimation of a probability density function and mode," Ann. Math. Stat., Vol. 33, September 1962, pp. 1065-1076.

[4] Poggio, Tomaso and Girosi, Federico, "Networks for Approximation and Learning", Proceedings of the IEEE, Vol. 78, No. 9, September 1991, pp. 1481-1497.

[5] Schoenberg, I. J., "Metric spaces and positive definite function," Ann. of Math, Vol. 44, 1938, pp. 522-536.

[6] Michelli,C. A., "Interpolation of scattered data: Distance matrices and conditionally positive definite functions," Constr. Approx., Vol. 2, 1986, pp. 11-22.

[7] Uncini, A.,Marchesi, M., Orlandi, G. and Piazza, F., "Improved Evoked Potential Estimation Using Neural Network", IEEE Conference on Neural Networks, Vol. I, San Diego, June 1990, pp.143-148.

[8] Hoyt, John D. and Wechsler, Harry, "An Examination of the Application of Multi-Layer Neural Networks to Audio Signal Processing", IEEE Conference on Neural Networks, Vol. II, San Diego, June 1990, pp. 305-310.

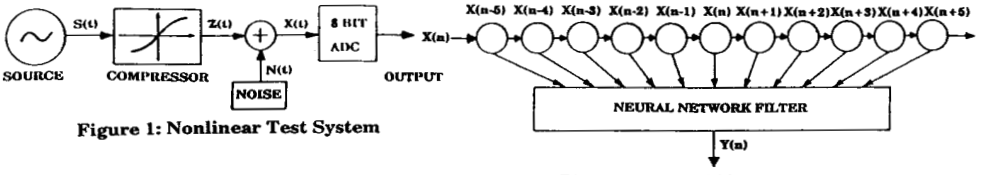

Figure 1: Nonlinear Test System

Figure 2: Neural Network Smoother

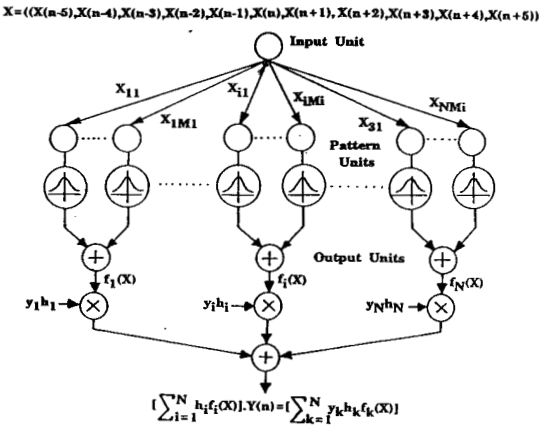

Figure 3: Probabilistic Neural Network (PNN)

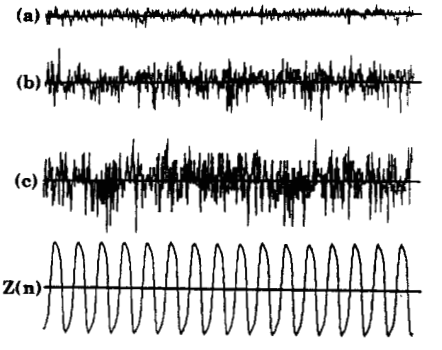

Figure 4: Noise N(n) & Compressed Signal Z(n)

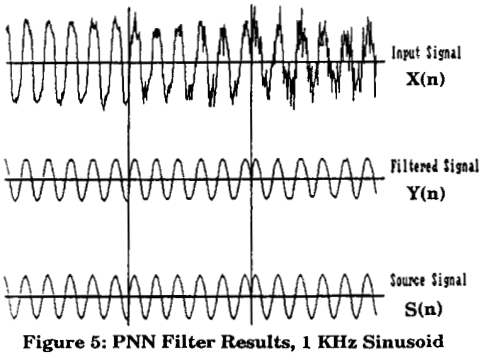

Figure 5: PNN Filter Results, 1 KHz Sinusoid

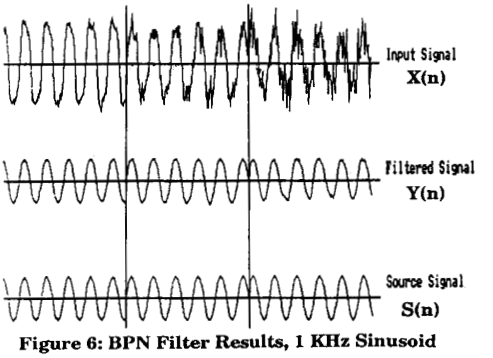

Figure 6: BPN Filter Results, 1 KHz Sinusoid

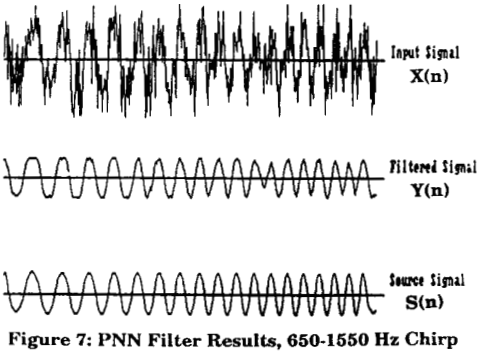

Figure 7: PNN Filter Results, 650-1550 Hz Chirp

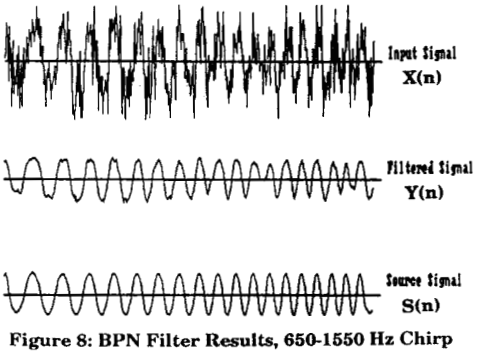

Figure 8: BPN Filter Results, 650-1550 Hz Chirp