Application of the Modified Probabilistic Neural Network to the Enhancement of Noisy Short Wave Radio Time and Morse Code Signals

Anthony Zaknich, Member IEEE, Associate Member of AES
Yianni Attikiouzel, Senior Member 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

Abstract

Nonlinear and linear techniques for the enhancement of short wave radio time and morse code signals corrupted by typical channel effects such as fading, random noise and impulse noise are investigated. Comparative results are given for solutions based on the Modified Probabilistic Neural Network, General Regression, Backpropagation neural networks and first second and third order Volterra Filters which demonstrate the advantages of the neural network approach to this type of problem.

On the Australian amplitude modulated (AM) short wave radio service there is a station which transmits an accurate time signal twenty four hours a day. The station call letters are VNG and it is broadcast on frequencies 16.000 MHz, 12.984 MHz and 8.638 MHz at power outputs of 5 KW, 3 KW and 10 KW respectively from an eastern states location [1]. The signal is a short tone burst every second with a longer one to signify the beginning of a minute. Other countries broadcast similar signals at various times usually on short wave frequencies very close to 5.000, 10.000, 15.000, 20.000 and 25.000 MHz. One such station from India with call sign ATA broadcasts at 10.000 MHz at a power output of 8 KW. Because these signals are broadcast over long distances at relatively low power outputs they are very susceptible to fading and various types of noises such as whistle caused by interference from station frequencies close by, white noise from receiving equipment and impulse noise from local electricity power systems. This makes them very hard to use for their intended purpose. A nonlinear filter design can solve this problem and recover the clean tone bursts. Morse code signals are still used for various purposes over various radio frequencies. These signals are fundamentally of the same type as the time signals and thus the same type of nonlinear filter can be used to enhance them as well.

This problem has a number of linear and nonlinear effects which are very difficult to deal with using linear techniques alone. The white noise could be reduced using a suitable linear bandpass filter, the fading could be handled using an automatic gain control (AGC) and the whistle could possibly be eliminated using a standard adaptive filter. However, impulse noise having a random amplitude and time of occurrence cannot be removed effectively with any linear or standard adaptive filtering techniques. Even using a narrow bandpass filter with AGC poses problems because there is a limit as to how narrow the filter can be made before the signal rise time and delay are increased beyond the limits of acceptable temporal precision. A nonlinear neural network filter or nonlinear vector mapping offers a solution to all of the above mentioned problems. It offers particular advantage in temporal resolution since it can be designed to have negligible delay. The output of the mapping can be defined as a scalar value y representing a filtered output point for an input vector x which consists of a digitally sampled portion of the unfiltered input time series signal. This is described later.

The Modified Probabilistic Neural Network (MPNN) [2,3] which is closely related to Specht’s General Regression Neural Network (GRNN) [4] offers an excellent solution to this problem.

\[
\hat{y}(x) = \sum_{i=1}^{NS} \frac{y_i \exp\left(-\frac{(x - x_i)^\top (x - x_i)}{2\sigma^2}\right)}{\sum_{i=1}^{NS} \exp\left(-\frac{(x - x_i)^\top (x - x_i)}{2\sigma^2}\right)}
\]  

(1)
If the $y_i$ are allowed to be individual real valued scalars, equation (1) becomes exactly Specht's GRNN which incorporates each and every training vector pair $(x_i, y_i)$ into its architecture ($x_i$ is a single training vector in the input space and $y_i$ is the associated desired scalar output). If it can be assume that there is only one centre in the input space per output $y_i$ then a convenient general model to use for all forms of the MPNN and the GRNN is:

$$
\hat{y}(x) = \sum_{i=1}^{M} \left( \frac{Z_i y_i \exp \left( \frac{-(x - centx_i)^T (x - centx_i)}{2\sigma^2} \right)}{2\sigma^2} \right)
$$

$\hat{y}(x)$ is the centre or mean vector for class i in the input space (real valued or quantised).

$\sigma$ is the single learning or smoothing parameter chosen during network training.

$y_i$ is the output related to $centx_i$ (real valued or quantised).

$M$ is the number of unique centres i in the MPNN structure.

$Z_i$ is the number of input training vectors $x_j$ associated with $centx_i$.

$NS = \sum Z_i$ is total number of training vectors.

Equation (2) is derived from the GRNN equation (1) using the following approximation:

$$
Z_i \exp \left( \frac{-(x - centx_i)^T (x - centx_i)}{2\sigma^2} \right) = Z_i \sum_{j=1}^{M} \exp \left( \frac{(x - x_j)^T (x - x_j)}{2\sigma^2} \right)
$$

This is a reasonable approximation if the $x_i$ are close together in a relatively small local space and can be adequately represented by a single centre vector $centx_i$. The key to the practical application of the general MPNN equation (2) is related to the method of selection of the $y_i$ and the grouping of the associated input vectors in each class i. One solution to this selection and grouping for simple sinusoidal signals proposed by Zaknich et al [2] was to uniformly quantise the noiseless desired $y_i$, separately group the $y_i$ having positive and negative slopes in the waveform and associate them with the mean of the input vectors mapping to each group. This simple case led to a more general approach, for both simple and more complex signals, of uniquely identifying the quantised $y_i$ having a similar local waveform pattern which is called the MPNN Method A [5]. Method A involved taking the desired waveform $y(t)$ and uniformly sampling it in time to $y(n)$ digital sample points which were then uniformly quantised into one of $N$ quantisation levels to be able to define a desired output phase state vector composed of $y(n)$ and the $m$-1 quantised samples immediately preceding it in time, i.e. $(y(n), y(n-1), ..., y(n-m-1))$. The greater the $m$ the more uniquely a quantised output value $y(n)$ or $y_i$ could be identified in the waveform for the purpose of associating all the input vectors $x_i$ mapping into the same output phase state. In most applications it was sufficient to use $m = 1$ with the $y(n)$ sample quantised to one of $N$ uniform levels and the $(y(n-1)$ to $N_s$ levels (usually $N = N_s$ but not necessarily). A number of other methods are discussed in [5] but only the further MPNN Method B was considered generally useful and practical to mention. It involved uniquely identifying the quantised $y_i$ associated with a local region of the input vector space defined by uniformly sampling and uniformly quantising the noisy input vector elements. This latter method reduced to an efficient realisation of a quantised version of the GRNN.

To compare the effectiveness of the various filter designs, sets of training and testing data were simulated to represent the generic type of signals and noise sources described above. Two independent signals each having 3000 points at a sampling frequency of 5 KHz (0.6 seconds) were constructed as follows. There were 8 tone bursts at a frequency of 400 Hz. Each burst had a random starting phase between 0 to $2\pi$ and was 188 points long (37.6 ms) beginning at every 375 points (75 ms). The equation for the sine wave was as follows: 

$$
[1.0 * \sin((2\pi 400 \text{ n} / 5000) + \theta)]
$$

where $\theta$ is a random phase and $n$ is the sampling index number. The whole signal of 3000 points was then modulated by $[0.25 * \cos(2\pi 13.3333 \text{ n} / 5000) + 0.75]$ to simulate a fading effect over the time period of each and every tone. To this was added random wide band noise with zero mean and a flat probability density function (pdf) between -0.33 to +0.33 plus 99 impulses. The impulses were located at every 30 + $r$ points ( $r$ is a random number 0 to 30) and had a random positive amplitude of between 0.5 to 2.0 and a length of 5 sample points. They all had a positive peak followed immediately by a negative peak with a magnitude value 20% of the positive one. The first set of the signals was the training sequence and the second the testing sequence as shown in Figure 1. It was arbitrarily
decided to adopt an input vector size $p=11$ and a smoother design as follows. Each discrete output sample point $y(n)$ mapped from an 11 element input vector $x$ was made up of the 5 discrete time series sample points immediately prior to and 5 following the current discrete sample point $x(n)$ of the digitally sampled nonlinear input signal, i.e.

$$x = [x(n-5), x(n-4), x(n-3), x(n-1), x(n), x(n+1), x(n+2), x(n+3), x(n+4), x(n+5)]^T,$$

vector.

The training and testing vector sets used in this investigation were as follows:

Training data - 3000 points, input dimension $p = 11$, fixed delay = 5 points.
Testing data - 3000 points, input dimension $p = 11$, fixed delay = 5 points.

In a real implementation it would be advisable to prefilter the input signal with a wide bandpass filter centred at 400 Hz to remove much of the out of band noise before applying it to the nonlinear filter. This was not done in this instance in order to test the comparative methods more effectively under harsher conditions.

The nonlinear and linear filters that were tested and applied to this problem included the GRNN equation (1), the first, second and third order Volterra filters, a BPN filter and the MPNN Methods A and B. The results are shown in Tables 1 and 2 and Figures 2 to 9. Quoted times are the running times of software implementations written in Borland C and executed on an 80386 AT compatible PC with a clock frequency of 33 MHz. The training times for the GRNN and the MPNN's were the times for one pass of the whole testing set whereas the times for the other filter networks were the total times to adequate convergence for the specified architecture.

The GRNN results are shown in Table 1 and Figure 2. Results for the best fit, in the least mean squared error (mse) sense, for the linear First Order Volterra Filter (linear finite impulse response filter), nonlinear Second Order (Quadratic) and Third Order Volterra filters [6] having eleven input nodes and one output node are shown in Table 1 and Figures 3 to 5. The Third Order Volterra filter was a little more effective than the Second Order Volterra filter which was a little more effective than the First Order Volterra filter but none were able to effectively capture the signal fading effect or adequately filter the impulse noise. The results for a BPN filter with 11 nodes in the first layer, 51 in the second, 21 in the third and 1 in the output are shown in Table 1 and Figure 6.

The Australian Journal of Intelligent Information Processing Systems

Spring 1995
Clearly, the GRNN had the best signal filtering capability but the GRNN network size was much larger than the others. Application of the MPNN Method A was able to considerably reduce the network size M but its performance was not as good as for the GRNN, as can be seen in Table 2 and Figure 7. On the surface this seemed surprising since this poor performance of the MPNN Method A did not occur when the same data as shown in Figure 1 but without the gaps (i.e. continuous sinusoid) was used. To gain insight into this problem it is instructive to view the data graphically in Figures 10 and 11.

Figure 10 shows the VNG data desired output 2-D phase plot for quantisations N = 256 and \( N_k = 256 \) overlayed with the input vector scatter plot. The characteristic elliptic shape associated with the output sine wave of constant amplitude is plotted with small filled squares. Each of these squares on the plot represents a number of points which fall into this quantised square region. The scatter plot of individual points represents the corrupted and noisy input signal vector points where only the first two dimensions are seen. The maximum input signal amplitude has been normalised to 1.0 for display purposes (it is actually 2.0 according to the synthesis equation given previously). The input signal points are scattered somewhat randomly due to the addition of the random noise and impulse noise. Figure 11 shows the scatter plot of the input signal during only the gap periods between sine bursts. The centre group of points which gives the impression of a square is due to the random noise having a flat pdf. The two rectangular extensions are due to the impulse noise. If this plot were to be subtracted from Figure 10 a scatter plot representing the input signal associated with the corrupted input sine signal alone would be seen.

A problem occurred with the MPNN Method A, especially for low values of \( N_k \) (\( N_k \) is the number of input centres allowed for each output class, \( N_k = 1 \) has been assumed in equation (2)), because the number of input clusters were insufficient to properly represent the large spread of the input vectors due to the impulse noise. In the gap regions, if \( N_k = 1 \), the single input centre represented the random noise very well but almost completely ignored the effect of the large excursions of the impulse noise. Consequently, a large impulse input signal from the signal gap region was forced to relate to input centres associated with the sine wave and thus mapped to the wrong output value. This effect can be seen very clearly in the MPNN filtered signal shown in Figure 7 where large impulses in the middle of the first gap look like parts of a sine signal rather than a zero level. A convenient way to fix this problem is to let the single centre, weighted by the number of vectors it replaced, represent the bulk of the
input space near it according to equation (2), and
then supplement this with extra centres of
individual input vectors which were sufficiently
distant from the centre. A normalised $\sigma_{\text{max}}$
distance away from the centre was defined. Points
within the distance were averaged to provide the
centre point according to equation (3) and those
outside were left as individual vectors. This
process was iterated several times until a stable
centre was achieved which represented the most
dense region. Some test results for various choices
of $\sigma_{\text{max}}$ are shown in Table 2. The process was
iterated 10 times in each case to ensure stability.

As $\sigma_{\text{max}}$ was reduced the MPNN Auxiliary
Method A performance approached that of the
GRNN. A $\sigma_{\text{max}} = 0.8$ gave a good result close to
that of the GRNN, as can be seen in Figure 8. This
method proved to be an effective means of
reducing training vectors for the GRNN. In
practice there is need for an upper limit to be
placed on the number of auxiliary single vectors
per centre to keep the network size finite for cases
of very large numbers of training vectors.

Another very effective way to reduce the GRNN
was via MPNN Method B. The input vectors were
quantised into $N_q$ levels per dimension and the
output was quantised to $N$ levels as usual. The
input vector quantisation was only used for
characterisation purposes, ie. the input centres
were derived by averaging the original input
unquantised vectors falling into each of the input
space quantised boxes. The input vectors in each p-
dimensional quantised box were grouped only
if they mapped into the same quantised output value.
In this way a total of $M$ unique centres resulted in
the input space which mapped into the correct
quantised output values. The results for a number
of different quantisation selections are shown in
Table 2 and the best filter output is shown in
Figure 9. For $N_q = N = \infty$ the MPNN was the same as
the GRNN since each input vector was associated
with its desired output.

Conclusions

MPNN Method B was more efficient and more
accurate than the previous Auxiliary Method A
and it ensured a finite network since all vectors
were forced to fall within a finite number of
quantised boxes. It was also computationally
simpler as it required no iterations to ensure
stability of the centres. As can be seen by the mse
results in Tables 1 and 2 the filtered waveform was
smoother and more accurate ($\text{mse} = 0.007856$)
than that for the MPNN Auxiliary Method A ($\text{mse} = 0.008630$) and the GRNN ($\text{mse} = 0.008464$)
which in turn were better than the BPN ($\text{mse} = 0.008711$) which was better than any of the
Volterra filters (third order $\text{mse} = 0.001301$, 
second order $\text{mse} = 0.01451$, first order $\text{mse} = 0.020352$). Although the BPN filter gave a
reasonably low mse it was not as effective at
reducing the impulse noise and it took an
extremely long time to train. The training times for
all the MPNN forms including the GRNN were
quite low compared to the BPN and the nonlinear
Volterra filters which relied on a gradient descent
training mechanism. All things considered,
Method B was the best solution to the problem
even though the execution time for a software
implementation was high compared to the fastest
BPN filter. The execution times on a PC however,
are irrelevant to real-time parallel
implementations.

References

[1] Sennitt, A. G., Editor, "World Radio TV
60.

[2] 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

[3] Zaknich, Anthony and Attikiouzel,
Yianni, "Automatic optimization of the
modified probabilistic neural network for
pattern recognition and time series
analysis", Proceedings of the First
Australian and New Zealand Conference
on Intelligent Information Systems, Perth,
Western Australia, 1-3rd December,
1993, pp. 152-156.

neural network", IEEE Transactions on
Neural Networks, Vol. 2, No. 6,

[5] Zaknich, Anthony and Attikiouzel,
Yianni, "Time Series Characterisation
Schemes for the Modified Probabilistic
Neural Network", Australian Journal of
Intelligent Information Processing
1-11.

[6] Lau, S. M., Leung, S. H. and Chan, B. L.,
"A reduced rank second-order adaptive
Volterra filter", ISSPA 92, Signal
Processing and Its Applications, Gold
Coast, Australia, August 16-21, 1992, pp.
561-563.
Table 1

<table>
<thead>
<tr>
<th>Filter Type</th>
<th>Data Set</th>
<th>Iterations</th>
<th>mse</th>
<th>σopt</th>
<th>Training time in seconds</th>
<th>Execution per vector in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRNN</td>
<td>Testing</td>
<td>-</td>
<td>0.008464</td>
<td>0.273</td>
<td>2,343.3</td>
<td>0.7811</td>
</tr>
<tr>
<td>1st Order Volterra</td>
<td>Training</td>
<td>250,000</td>
<td>0.022834</td>
<td>-</td>
<td>165.9</td>
<td>0.0003</td>
</tr>
<tr>
<td></td>
<td>Testing</td>
<td></td>
<td>0.020352</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2nd Order Volterra</td>
<td>Training</td>
<td>5,850,000</td>
<td>0.015700</td>
<td>-</td>
<td>19,144.0</td>
<td>0.0013</td>
</tr>
<tr>
<td></td>
<td>Testing</td>
<td></td>
<td>0.014510</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3rd Order Volterra</td>
<td>Training</td>
<td>550,000</td>
<td>0.001316</td>
<td>-</td>
<td>23,948.0</td>
<td>0.0177</td>
</tr>
<tr>
<td></td>
<td>Testing</td>
<td></td>
<td>0.001301</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BPN</td>
<td>Training</td>
<td>8,200,000</td>
<td>0.005386</td>
<td>-</td>
<td>721,076.0</td>
<td>0.0265</td>
</tr>
<tr>
<td></td>
<td>Testing</td>
<td></td>
<td>0.008711</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2

<table>
<thead>
<tr>
<th>Filter Type</th>
<th>M</th>
<th>N</th>
<th>Ns</th>
<th>σmax</th>
<th>mse</th>
<th>σopt</th>
<th>Training time in seconds</th>
<th>Execution per vector in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPNN, Method A</td>
<td>190</td>
<td>256</td>
<td>256</td>
<td>$\infty$</td>
<td>0.014480</td>
<td>0.407</td>
<td>139.3</td>
<td>0.0461</td>
</tr>
<tr>
<td>MPNN, Auxiliary Method A</td>
<td>1,668</td>
<td>256</td>
<td>256</td>
<td>0.8</td>
<td>0.008630</td>
<td>0.286</td>
<td>1,305.0</td>
<td>0.4350</td>
</tr>
<tr>
<td></td>
<td>1,198</td>
<td>256</td>
<td>256</td>
<td>1.0</td>
<td>0.009213</td>
<td>0.298</td>
<td>935.1</td>
<td>0.3117</td>
</tr>
<tr>
<td></td>
<td>774</td>
<td>256</td>
<td>256</td>
<td>1.2</td>
<td>0.009772</td>
<td>0.323</td>
<td>604.2</td>
<td>0.2104</td>
</tr>
<tr>
<td></td>
<td>545</td>
<td>256</td>
<td>256</td>
<td>1.5</td>
<td>0.010223</td>
<td>0.335</td>
<td>424.4</td>
<td>0.1418</td>
</tr>
<tr>
<td></td>
<td>288</td>
<td>256</td>
<td>256</td>
<td>1.8</td>
<td>0.010260</td>
<td>0.336</td>
<td>209.6</td>
<td>0.0699</td>
</tr>
<tr>
<td></td>
<td>211</td>
<td>256</td>
<td>256</td>
<td>2.0</td>
<td>0.011618</td>
<td>0.347</td>
<td>153.6</td>
<td>0.0512</td>
</tr>
<tr>
<td>MPNN, Method B</td>
<td>3,000</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>0.008464</td>
<td>0.273</td>
<td>2343.3</td>
<td>0.7811</td>
</tr>
<tr>
<td></td>
<td>1,407</td>
<td>4</td>
<td>64</td>
<td>$\infty$</td>
<td>0.007856</td>
<td>0.283</td>
<td>1118.7</td>
<td>0.3729</td>
</tr>
<tr>
<td></td>
<td>1,359</td>
<td>4</td>
<td>32</td>
<td>$\infty$</td>
<td>0.007894</td>
<td>0.285</td>
<td>1080.9</td>
<td>0.3603</td>
</tr>
<tr>
<td></td>
<td>586</td>
<td>2</td>
<td>64</td>
<td>$\infty$</td>
<td>0.009294</td>
<td>0.296</td>
<td>465.0</td>
<td>0.1550</td>
</tr>
</tbody>
</table>