EXTRACTION OF A SPEECH SIGNAL IN THE PRESENCE OF A MUSICAL NOTE SIGNAL

K. Chong and R. Togneri

Centre for Intelligent Information Processing Systems
Department of Electrical and Electronic Engineering
The University of Western Australia

ABSTRACT - This paper presents the methodology of extracting a speech signal in the presence of a musical note signal using the GRNN (General Regression Neural Network). An overview of GRNN is presented first, followed by preliminary simulations. Results of extracting speech in the presence of a flute and a cello note are also presented.

INTRODUCTION

Speech and musical note signals have overlapping spectra. Hence, extraction of the speech signal cannot be done by using conventional filters. The new approach is to employ a type of artificial neural network (ANN) called the General Regression Neural Network (GRNN). The GRNN does not need any information from the frequency domain – it learns to map an input signal to the desired output signal by way of examples. The methodology involves training the GRNN to map an incoming signal, which is a composite of speech and musical note signals, to the desired output signal, which is the musical note signal alone. If the mapping is done successfully, then the speech signal can be extracted by subtracting the output signal from the input signal.

OVERVIEW OF GENERAL REGRESSION NEURAL NETWORK

The Regression Function Concept

Before discussing the regression function concept, it is worthwhile to clarify the notations used. The standard notation is adopted where upper-case letters are used to denote a random variable while lower-case letters are used to denote a specific value. For example, X is the random variable and x is a value. Hence, $X = x$ means that the random variable X takes on the value x.

A regression function performed on an independent variable X computes the most probable dependent variable Y, based on a finite number of noisy measurements of X and the associated values of Y. The variables X and Y can be thought of as the input and the output of a system respectively. The regression function could be a mathematical function of X with unknown parameters, a_i. In the case of linear regression, for example, the output Y is a linear function of the input X. The unknown parameters a_i are hence the linear coefficients.

A General Regression Neural Network

The General Regression Neural Network (GRNN) was first introduced by Donald F. Specht (Specht, 1991). It performs a regression of Y on X. The method employed is not governed by a specific function. Rather, it allows the appropriate form to be expressed as a probability density function (pdf) that is empirically determined from the observed data using Parzen window estimation (Parzen, 1962).

The variables X and Y can be vectors. For the rest of this paper, the input variable considered will be a vector while the output variable a scalar. The notation used for the remainder of this paper is the unput vector $\boldsymbol{X}$ ($\boldsymbol{X} = [X_1\ X_2 \cdots X_p]^T$ where T denotes transpose) and scaler output Y.

The regression performed by the GRNN is in fact the conditional expectation of Y given $\boldsymbol{X} = \boldsymbol{x}$. In other words, it outputs the most probable scalar Y given a specified input vector $\boldsymbol{x}$. Mathematically, it is expressed as

$$E[Y|x] = \int_{-\infty}^{\infty} y f_Y(y|x)dy = \frac{\int_{-\infty}^{\infty} y f_{XY}(x,y)dy}{\int_{-\infty}^{\infty} f_{XY}(x,y)dy} \tag{1}$$

From equation (1), it is evident that if the joint pdf of X and Y is pre-determined, then the expected value can be computed. Otherwise, it must usually be estimated from a sample of observations of X and corresponding Y using non-parametric estimators.

A suitable estimator, $\hat{f}_{XY}(x,y)$, of $f_{XY}(x,y)$ is given below.

$$\hat{f}_{XY}(x,y) = \frac{1}{(2\pi)^{(p+1)/2}\sigma^{(p+1)}}\frac{1}{n}\sum_{i=1}^{n}\exp\left\{-\frac{(x-\overline{x}_i)^T(x-\overline{x}_i)}{2\sigma^2}\right\}\exp\left\{-\frac{(y-\overline{y}_i)^2}{2\sigma^2}\right\} \tag{2}$$

where $\overline{y}_i$ is the desired scalar output given the observed input vector $\overline{x}_i$
p is the dimension of each $\overline{x}_i$ and x
n is the number of spherical Gaussian pdfs in the summation
σ is the standard deviation (also referred to as the smoothing factor)
T denotes transpose

Substituting equation (2) into equation (1) and performing the indicated integrations yield:

$$\hat{E}[Y|X = x] = \frac{\sum_{i=1}^{n}\overline{y}_i\exp\left\{-\frac{R_i^2}{2\sigma^2}\right\}}{\sum_{i=1}^{n}\exp\left\{-\frac{R_i^2}{2\sigma^2}\right\}} \tag{3}$$

where $R_i^2 = (x-\overline{x}_i)^T(x-\overline{x}_i)$

In equation (3), the parameters are $\overline{x}_i$, $\overline{y}_i$, and σ. The $\overline{x}_i$ and $\overline{y}_i$ are directly determined from observing inputs and desired outputs (thus n is the number of observations). Each $(\overline{x}_i, \overline{y}_i)$ pair is a training input vector, output scalar pair, and n of these pairs form the training set stored in the local memory. By this conceptualisation, the GRNN can be thought of as a memory-based network that provides estimates of continuous variables, and converges to the underlying regression surface. The regression surface can be linear or non-linear.

When the underlying parent distribution is not known, an optimum σ cannot be determined for a given number of observations n. The method used to find an appropriate σ is therefore empirical rather than theoretical. The process for locating the most suitable σ is termed training.

There are two ways in training the GRNN. One way is to present to the network another independent set of input-output observations called the testing set. Using equation (3), the GRNN computes outputs repeatedly using known (and fixed) parameters $\overline{x}_i$ and $\overline{y}_i$ (in the training set), and a varying σ. For each σ, the computed outputs are compared to the desired outputs found in the testing set. The performance of the GRNN using a particular σ is evaluated using the mean-squared error (MSE), given as:

$$\text{MSE} = \sum_{i=1}^{m} (y_{o,i} - y_{d,i})^2 \tag{5}$$

where $y_{o,i}$ is the i^{th} actual output (corresponding to the i^{th} input vector in the testing set)
$y_{d,i}$ is the i^{th} desired output found in the testing set
m is the number of observations in the testing set

The most appropriate σ is the one corresponding to the least MSE value.

If the testing set is not available, another method termed the holdout-one test can be employed to locate the most suitable σ. In this case, one vector from the training set is taken out at a time and tested against the remainder vectors. Hence at any instance, there should only be n – 1 training vectors in the memory since one is taken out to be used as a testing vector. As before, σ varies for each complete test, and the MSE is used to evaluate the performance of each σ. The most suitable σ would be the one corresponding to the least MSE value.

How the GRNN is Applied to Digital Signal Processing

When applying the GRNN to digital signal processing, it is essential to convert the incoming digital signal to a sequence of input vectors. The elements of the first vector are the first p samples of the incoming signal, where p is the input vector dimension. The next vector would contain samples that are advanced by one sample from the last vector. This advancement of the input vector frame continues until all the samples in the input signal are contained in at least one input vector. Hence for a signal with N samples, N-p+1 input vectors can be constructed. Note that the number of output values is also N-p+1. Therefore, after passing through the GRNN, the filtered signal will lose p–1 samples.

PRELIMINARY SIMULATIONS

Before the GRNN was applied to extract speech from the presence of a musical note signal, a preliminary simulation was done. The purpose of this simulation was to study the behaviour of the GRNN under basic signal processing operations.

The simulation discussed here is that of filtering a harmonic combination in the presence of Gaussian noise. The equation for the original signal used is given by sin(2πft) + sin(4πft) + sin(6πft) + sin(8πft). This signal is shown in figure 1. The Gaussian noise is characterised by zero mean and a standard deviation of 0.5. The overall corrupted signal is shown in figure 2.

The best input vector dimension for this problem was 70, the most appropriate number of training pairs was 750, and the most suitable σ was found to be 0.90. The corresponding MSE was 0.0038410402. After the training was completed, a similar type of signal as that shown in figure 2 was presented to the GRNN for filtering. The resulting output signal, having lost the first 69 samples, is shown in figure 3.

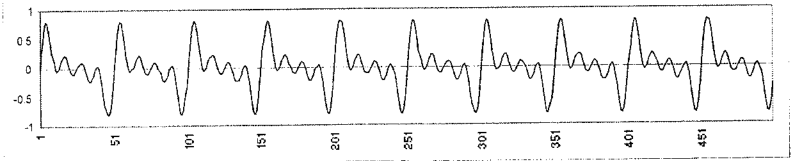

Figure 1 Original Harmonics of Sinusoidal Signals

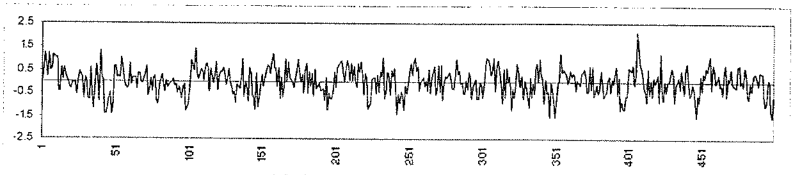

Figure 2 Corrupted Signal

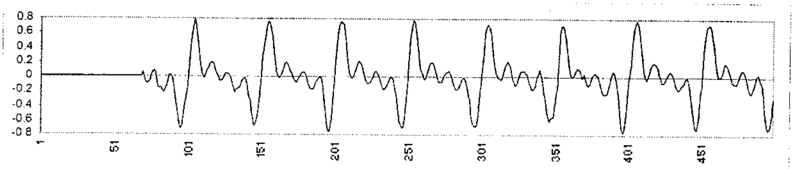

Figure 3 Output Signal

Comparing figure 3 with 1 shows that the waveforms are identical except that the amplitude of the output signal has dampened. This is referred to as the amplitude dampening effect.

EXTRACTION OF SPEECH

In the extraction process, the function of the GRNN is to filter off the speech signal from an input signal which consists of both the speech and the musical note signals. If the output signal is exactly the original musical note signal, then the speech signal can be reconstructed exactly by subtracting the output signal from the input signal. The reason for this approach is because the GRNN is more capable of filtering off the irregularities (speech signal) and retaining the periodic (musical note signal).

The speech signal employed for the extraction problem, shown in figure 4, is a male voice saying "this is a speech signal". The corrupting musical note signal is a flute note at frequency 440 *Hz*. Both signals were sampled at 8000 *Hz*. The reason for the chosen sampling rate is to minimise the number of observations and hence computations and yet maintain the fidelity of the signals.

The samples of the digitised musical note were scaled to within -0.9 and 0.9, and the digitised speech signal to within -0.1 and 0.1. The corrupted signal is an additive combination of the two scaled signals, as shown in figure 5. As the result of the corruption, the words in the speech are no longer recognisable.

The GRNN was trained to filter off the speech from the corrupted signal (figure 5). The most suitable σ found was 0.034 with an MSE value of 0.0000363579. After the signal in figure 5 was passed through the GRNN, the output signal was subtracted from the input signal. The resulting signal is shown in figure 6. The words in this filtered speech signal are now recognisable. However, a soft flute tone can be heard in the background (evident in figure 6), this can be explained by the amplitude dampening effect. The output signal from the GRNN (before it was subtracted from the input) had its amplitude dampened. When subtracting this dampened signal from the input signal, an error signal (heard as a flute tone) would result, and this error signal is imposed on the filtered speech signal.

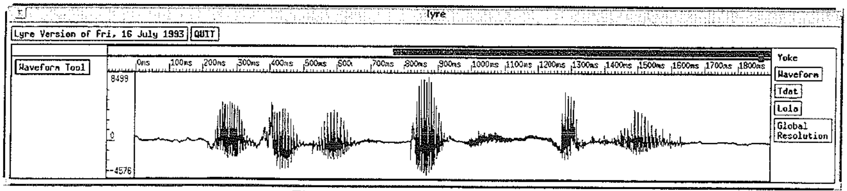

Figure 4 Original Speech Signal

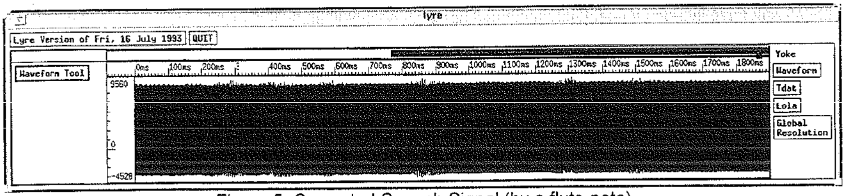

Figure 5 Corrupted Speech Signal (by a flute note)

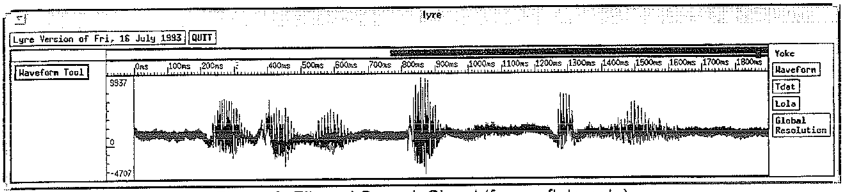

Figure 6 Filtered Speech Signal (from a flute note)

A cello note was used next in the place of the flute note. The cello note was of frequency 440 *Hz* and sampled at 8000 *Hz*. The same corruption level was used. The corrupted speech signal is shown in figure 7. Due to the rich quality tone of a cello note, not even the presence of the speech signal was evident. After the GRNN was trained successfully, the most suitable σ found was 0.62 with MSE value of 0.0012753863. Next, the signal in figure 7 was passed through the GRNN, and the output signal was then subtracted from the input signal. This subtracted signal (filtered speech signal) is shown in figure 8. Although there is a strong tone in the background, the words in the filtered speech signal are recognisable. The reason for the presence of this strong tone is once again, the error signal being imposed on the filtered speech signal.

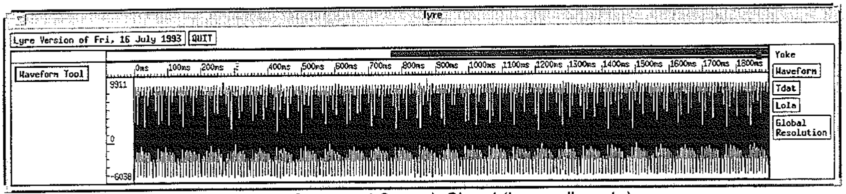

Figure 7 Corrupted Speech Signal (by a cello note)

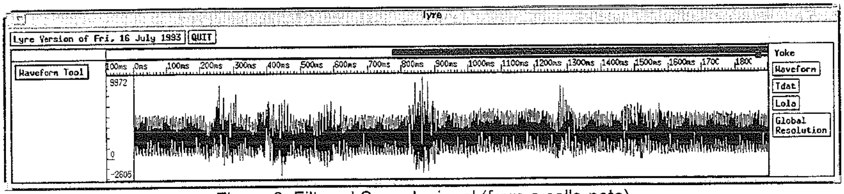

Figure 8 Filtered Speech signal (from a cello note)

CONCLUSION

It has been shown that the GRNN does not need any information about the frequency domain characteristics of the signals to be filtered. It maps signals to one another by way of examples. Hence it has the ability to separate a speech signal and a musical note signal although they have overlapping spectra. However, the current form of the GRNN does suffer from some limitations. One limitation is that it needs computer memory to store its training set. Another limitation is in its inability to filter signals in real time. Both of these limitations can be overcome by performing a prior clustering of the training observation sequences and thereby reduce the number of observation samples used to form the network. Another problem is the amplitude dampening effect, however since this is a gain effect some form of optimal gain control can be used.

REFERENCES

Dayhoff, J.E. (1990) *Neural network architectures: an introduction*, (Van Nostrand Reinhold: New York).

Parzen, E. (1962) *On estimate of a probability density function and mode*, Ann. Math. Stat., Vol. 33, pp. 1065-1076.

Specht, D.F. (1991) *A general regression neural network*, IEEE Transactions on Neural Networks, Vol. 2, No. 6, pp. 568-576.

Widrow, B. and Winter, R. (1988) *Neural networks for adaptive filtering and adaptive pattern recognition*, IEEE Computer, pp. 25-39.

Zaknich, A., deSilva, C. and Attikiouzel, Y. (1991) *Time series characterisation schemes for the modified probabilistic neural network*, Australian Journal of Intelligent Information Processing Systems (AJIIPS), Vol. 2, No. 2, pp. 1-11.

Zaknich, A., Attikiouzel, Y. (1995) *Application of artificial neural networks to nonlinear signal processing*, IEEE Press, Chapter 21 in "Computational intelligence: a dynamic systems perspective", ISBM 0-7803-1182-5 pp. 292-308.