Volume 2, Issue 2, February 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Filtering Noises from speech Signal : A BPNN approach

Debananda Padhi, Brataati Mahaanty, Sucharita Padhi

Master of Engg., CSE (KE)
P.G. Dept. of CS&A, Utkal University
Vani Vihar, Bhubaneswar, Odisha
Debananda.padhi106@gmail.com

***Abstract:* This paper gives a nural network of the type feed forward error Back Propagation (BPNN) which is used for filtering of noise. The purpose of this paper was to gain insight into the way neural network can be trained to remove noise from a noise corrupted signal with implications for the signal processing in general. The neural network simulation was implemented in MATLAB. Training of the neural network occurred on a set of spectral data with random transitions and line shape parameter. This paper represents mathematical representation of BPNN and noise filtering using error back propagation.**

Keywords: BPNN; Adaptive filter; Noise signal;

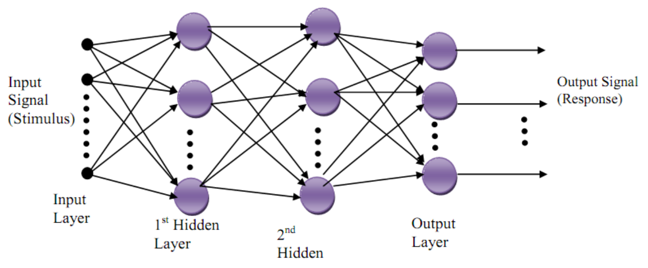
I. INTRODUCTION

Filtering is an important and pervasive problem that is found in many applications such as transmission of speech in a noisy environment and the reception of data in a noisy channel. Filter is applied to a set of noisy data, in order to extract information. In many speech communication applications such as hands free mobile telephony, hearing aids and voice controlled system the recorded speech signal are often corrupted by background noise. Background noise is broadband and non stationary and the SNR of microphone signals may quite low. Background noise causes a signal degradation that can lead to totally unintelligibility of the speech signal and decreases the performance of speech coding and speech recognition. so speech enhancement or noise reduction is done . For example, in an airplane, When the pilot speaks into a microphone, the engine noise in the cockpit is added to the voice signal, and the resultant signal heard by passengers would be of low quality. We would like to obtain a signal that contains the pilot's voice, but not the engine noise. We can do this with an adaptive filter if we obtain a sample of the engine noise and apply it as the input to the adaptive filter. The BP algorithm consists of at least three layers of simulated neurons, called input, hidden (middle), and output layer. All neurons are connected to the neurons of the previous and following layers, resembling the connectivity of neurons in the brain. Each connection is weighted and the weights are successively adjusted to optimize the network. This is achieved by comparing the neural network output with the desired output and propagating the error correction back through the network.

II ARTIFICIAL NEURAL NETWORK

Artificial Neural Network (ANN) is an information processing paradigm that is inspired from biological nervous systems, such as the brain process information. It is composed of a large number of highly interconnected processing elements (neurons) working in union to solve specific problem. Basically, neural networks are built from simple units, sometimes called neurons or cells. Artificial

Neural Network is information processing devices with the capability of performing computations similar to human brain or biological neural network. There are mainly three different types of layers presented in most ANNs. The first layer is called the input layer. Its main task is to receive input from the outside world. This layer has number of neuron equal to the number of model input. The layer next to the input layer is called the hidden layer. This layer is receiving input from the immediately preceding layers. The final layer of the network is called the output layer. The neuron present in this layer presents the output of the network. Neurons in any layer are fully connected to all neurons in the next layer.



The neurons in the same layer are not connected among each other. A weighted sum of the neuron inputs specifies the activation (i.e. sigmoid) function argument.

.

A . Architectures of artificial neural networks

The architecture of an artificial neural network is defined by the characteristics of a node and the characteristics of the node's connectivity in the network. The basic characteristics of a single node have been given in figure 2. Typically, network architecture is specified. by the number of inputs to the network, the number of outputs, the total number of elementary nodes that are usually equal processing elements for the entire network, and their organization and

Figure 1: A graph of a multi layered feed forward perception architecture with two hidden layers

interconnections. Neural networks are generally classified into two categories on the basis of the type of interconnections: *feed forward* and *recurrent*.

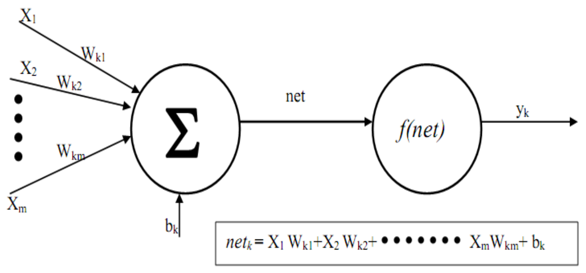


Figure 2: Architecture of single node of NN

B. *Feed forward approach*

The network is ***feed forward*** if the processing propagates from the input side to the output side unanimously, without any loops or feedbacks. In a layered representation of the feed forward neural network, there are no links between nodes in the same layer; outputs of nodes in a specific layer are always connected as inputs to nodes in succeeding layers. This representation is preferred because of its modularity, i.e., nodes in the same layer have the same functionality or generate the same level of abstraction about input vectors. Example as in figure 1.

C. *Back propagation*

"How does back propagation work?" Back propagation learns by iteratively processing a data set of training tuples, comparing the network's prediction for each tuple with the actual known target value. The target value may be the known class label of the training tuple (for classification problems) or a continuous value (for prediction). For each training tuple, the weights are modified so as to minimize the mean squared error between the network's prediction and the actual target value. These modifications are made in the "backwards" direction, that is from the output layer, through each hidden layer down to the first hidden layer (hence the name back propagation).
The steps are described below.
Algorithm: Backpropagation. Neural network learning for classification or prediction, using the backpropagation algorithm.
Input:
D, a data set consisting of the training tuples and their associated target values;
l, the learning rate;
network, a multilayer feed-forward network.
Output: A trained neural network.
Method:

```
(1) Initialize all weights and biases in network;
(2) while terminating condition is not satisfied {
(3) for each training tuple X in D {
(4) // Propagate the inputs forward:
(5) for each input layer unit j {
(6) $O_j = I_j$ ; // output of an input unit is its actual input value
(7) for each hidden or output layer unit j {
(8) $I j = \sum_i w_{ij} O_i + \theta_j$ ; //compute the net input of unit j with respect to the previous layer, i
(9) $Oj = 1/1+e^{-1}j$;}// compute the output of each unit j
(10) // Backpropagate the errors:
(11) for each unit j in the output layer
(12) $Err_j = O_j(1- O_j)(T_j - O_j)$; // compute the error
(13) for each unit j in the hidden layers, from the last to the first hidden layer
(14) $Err_j = O_j(1- O_j)\sum_k Err_k w_{jk}$; // compute the error with respect to the next higher layer, k
(15) for each weight wi j in network {
(16) $\Delta w_{ij} = (l)Err_j O_i$; // weight increment
(17) $w_{ij} = w_{ij} + \Delta w_{ij}$ ;} // weight update
(18) for each bias $\theta_j$ in network {
(19) $\Delta\theta_j = (l)Err_j$ ; // bias increment
(20) $\theta_j = \theta_j + \Delta\theta_j$ ;} // bias update
(21) }}
```

D. *Mathematical review of the BP neural network*

Every neuron of the input layer is connected to every neuron in the hidden layer. Similarly every neuron in the Hidden layer is connected with every neuron in the output layer. The connection is called weights. Number of neurons in the input layer is equal to the size of the input vector of the Neural Network. Similarly number of neurons in the output layer is equal to the size of the output vector of the Neural Network. Size of the hidden layer is optional and altered depends upon the requirement.
For every input vector, the corresponding output vector is computed as follows:
[hidden vector] = func ([Input vector]*[Weight Matrix 1] + [Bias1])
[output vector] = func ([hidden vector]*[Weight Matrix 2] + [Bias2])
The weight connecting the ith neuron in the first layer and jth neuron in the hidden layer is represented as Wij The weight connecting ith neuron in the hidden layer and jth neuron in the output layer is represented as Wij. The input vector is represented as [i1 i2 i3]. Hidden layer output is represented as [h1] and output vector is represented as [o1 o2]. The bias vector in the output layer is given as [b1 b2]. Desired output vector is represented is [t1 t2]
The vectors are related as follows.
h1=func1 (w11*i1+w21*i2+w31*i3 + bh)
o1= func2 (w11'*h1 +b1)
o2= func2 (w12'*h1 +b2)
t1~=o1
t2~=o2

E. *Multi-layered Neural Network*

For the present work, we have used a multi-layered neural network with two hidden layers each containing ten neurons. In addition, we have an input layer containing six neurons (one for each input) and a single output neuron which represents the character. The network is fully connected and its weights are updated using the gradient descent method i.e,

$$(\omega)^{new} = (\omega)^{old} - \eta\left(\frac{\partial E}{\partial \omega}\right) \quad ----(1)$$

Where $E = \pi r^2 \frac{1}{2} \sum_{i3} \left(y_{i3}^d - y_{i3}\right)^2$

The back propagated errors are calculated at each layer and using the generalized delta rule, the following weight update formulae are obtained for the network.

$\mathrm{W}_{i3,i2}(t+1) = \mathrm{W}_{i3,i2}(t) + \eta\delta_{i3}v_{i2}$ --- (2)

Where $\delta_{i3} = y_{i3}(1-y_{1_3})(y_{i3}^{d} - y_{i3})$

$\mathrm{W}_{i2,i1}(t+1) = \mathrm{W}_{i2,i1}(t) + \eta\delta_{i2}v_{i1}$ ----(3)

Where $\delta_{i2} = v_{i2}(1-v_{1_2})\sum_{i3}\delta_{i3}\mathrm{W}_{i3,i2}$

$\mathrm{W}_{i1,i0}(t+1) = \mathrm{W}_{i1,i0}(t) + \eta\delta_{i1}x_{i0}$ ---(4)

Where $\delta_{i1} = v_{i1}(1-v_{1_1})\sum_{i2}\delta_{i2}\mathrm{W}_{i2,i1}$

The output v each neuron is calculated using the sigmoidal activation function given by,

$v = \frac{1}{(1+e^{-h})}$ ---(5)

Where h is the original output for our network $i_0 = \{1,2,3,4,5,6\}$, $i_1 = \{1,2,3,\ldots,10\}$, $i_2 = \{1,2,3,\ldots,10\}$, $i_3 = \{1,2,3,\ldots,10\}$, the learning rate η is taken as 0.5.

II. ADAPTIVE FILTER

Adaptive filter theory is an important area in digital signal processing with many important applications like noise cancellation having been researched for more than four decades. Filters with changing coefficients is Adaptive filter. Most adaptive filter algorithms share the same common goals i.e. rapid convergence to a good approximation of the solution to the Wiener Hopf equation in a stationary environment, good tracking of the time-varying Wiener solution in non-stationary environments, and small filter coefficient deviations from the Wiener solution in a stationary environment after convergence. All these objectives shall be satisfied with algorithms characterized by the lowest possible computational complexity.

III. NOISE REMOVAL WITH A BPNN NETWORK

Back-propagation Neural Networks are the most famous neural type. The input of the network consisted of the signal plus the noise and the desired output, to perform the error correction to the neural network weights, was given by the noise-free signal. The dimension of the neural network layers were chosen to be 15 units in in- and output layers, while the size of the middle layer was varied to seek optimum performance. Each spectrum was piecewise presented to the neural net, 15 data points at the time, with immediate error back- propagation. During a training session each spectrum of **as,** the set was run repeatedly, here mostly in 50 iterations. Convergence of the system slightly improved when the order of spectra in the training set run through the BP.

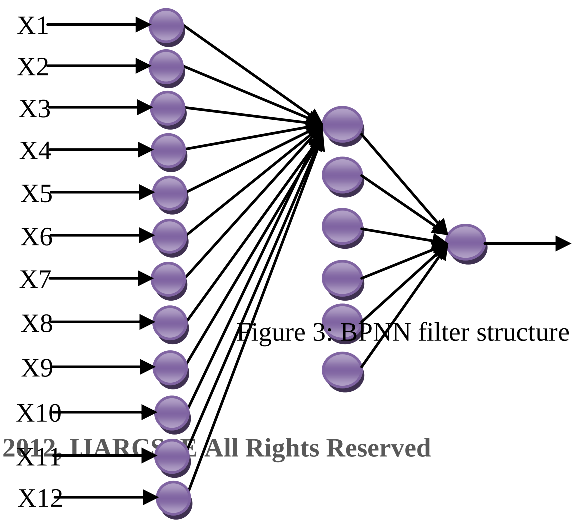


Figure 3: BPNN filter structure

IV. ALGORITHM

Step 1: Back propagation neural network is constructed with 12 input Neurons and 1 output neuron and 6 Hidden neurons

Step 2: In signal processing point of view, input of the neural network is the corrupted signal and the output of the neural network is the filtered signal.

Step 3: During the training stage, the elements of the Input vectors are the samples collected from the corrupted reference signal. Similarly element of the output vector is the corresponding sample collected from the reference signal.

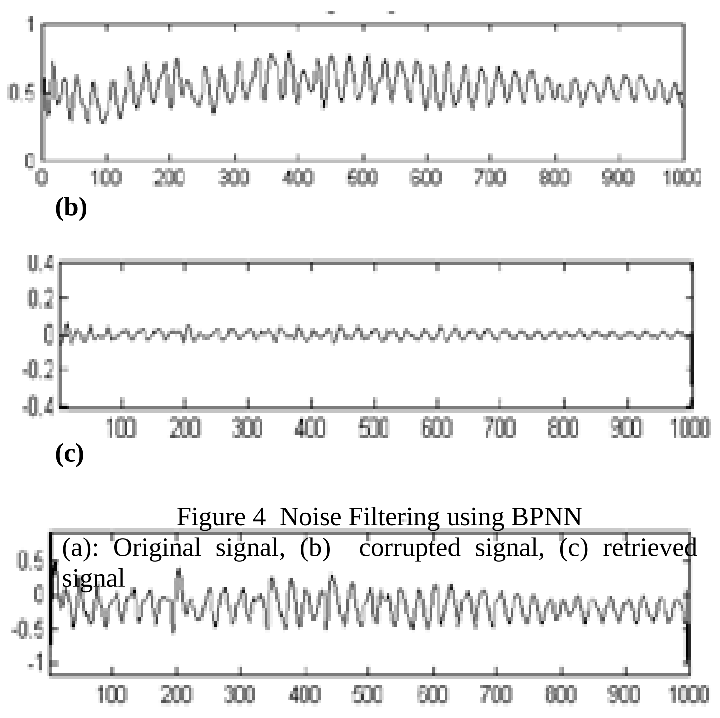
Step 4: Train the Artificial Back propagation Neural Network Store the Weights and Bias.

Step 5: The BPNN filter is ready to filter the original corrupted signal.

V. EXPERIMENTAL RESULTS

The low frequency information is lost and that is not retrieved in the filtered signal because of the fact that first 100 samples are used to train the network. If the training sequence is increased, low frequency information can be preserved in the filtered signal. Neural Network toolbox is used to train the network.

(a)



(b)

(c)

Figure 4 Noise Filtering using BPNN

(a): Original signal, (b) corrupted signal, (c) retrieved signal

VI. CONCLUSION

Filtering approach using error back propagation method for noise reduction is discussed. The result are shown using MATLAB. The noise is reduced using adaptive error back propagation.

VII. ACKNOWLEDGMENT

To complete this work, I have got valuable suggestions and guidance from different experts of this field. I thankful to all the contributors on this field from whose valuable reference I have taken for complete the work. The Book

VIII. REFERENCES

[1] P. Gorman and T. Sejnowski. “Analysis of hidden units in a layered network trained to classify sonar targets,” *Neural Networks,* vol. 1, 1988.

[2] D. Burr, “A neural network digit recognizer,” in *Proc. IEEE Int. Con$ Syst., Man, and Cybern.,* 1987.

[3] H. Bourlard and C. Wellekens. “Multilayer perceptrons and automatic speech recognition” ,*Proc. IEEE First Int. Conf Neural Networks*, 1987.

[4] M. Watson. “A neural network model for phenomena recognition using the Generalized Delta Rule for strength modification,” *Proc. IEEE First Int. Conf Neural Networks,* 1987.

[5] N. Dodd, “Multi-layer perceptrons inversion applied to image neighborhood operations.” *Proc. IEEE First Int. Conf: Neural Networks.* 1987.

[6] P. K. Simpson, *Artificial Neural Systems: Foundations, Paradigms, Applicarions, and Implementations.* Elmsford, NY: Pergamon, 1990

[7] Neural Network Toolbox by *Howard Demuth & Mark Beale* 8-Kisi, O., 2004. River flow modeling using artificial networks. ASCE J. Hydrol. Engg., vol 9: pp 60-63.

[9] Cigizoglu, H.K. and O. Kisi, “Flow prediction by three back-propagation techniquesusing k-fold partitioning of neural network training data” Nordic Hydrol., vol. 36.,2005

[10] M. Ibnkahla, “Applications of neural networks to digital communications: A survey,” *Signal Processing*, vol. 80, no. 7, pp.1185–1215, 2000

[11] Simon Haykin, “Adaptive Filter Theory, 4th Edition”, Pearson Education, Inc., 2002