Neural Networks For Signal Detection and Spectrum Estimation

Dr. Yogananda Isukapalli

SIGNAL DETECTION IN ADDITIVE NOISE

Detection of known Signals in Additive Noise

Observation Model

$X_i = \theta s_i + N_i$, $I = 1, 2, 3, \ldots, n$

$s = (s_1, s_2, \ldots\ s_n)^T$	:	Signal Vector
$N = (N_1, N_2, \ldots\ N_n)^T$	:	Noise Vector
$X = (X_1, X_2, \ldots\ X_n)^T$	:	Observation Vector

θ = Signal-strength Parameter

$$SNR = \frac{Signal\ Energy}{Noise\ Variance}$$

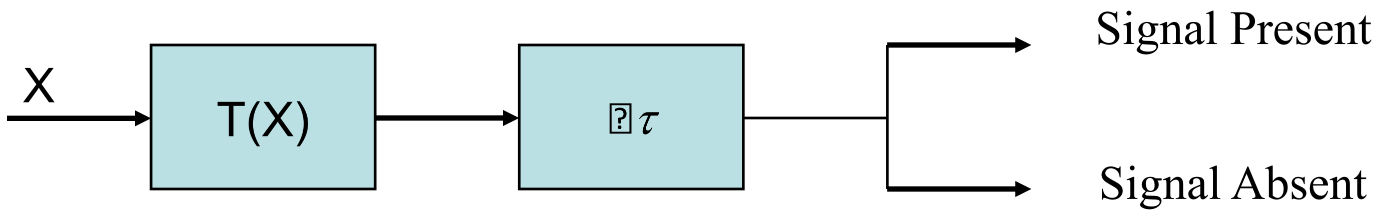
$$= \frac{\theta^2 \sum_{i=1}^{n} S_i^{\ 2}}{\sigma^2}$$

SIGNAL DETECTION IN ADDITIVE NOISE

Binary Test of Hypotheses

$H_0 : \theta = 0$ (Noise – only)

$H_1 : \theta > 0$ (Signal – plus – noise)



Performance Measures

Detection Probability

$$P_d = P[T(X) > \tau \mid H_1]$$

False Alarm Probability

$$P_d = P[T(X) > \tau \mid H_0]$$

SIGNAL DETECTION IN ADDITIVE NOISE

Matched Filter Detector

$$T_{MF}(X) = \sum_{i=1}^{n} s_i X_i$$ (White Noise)

- Uses linear test statistic
  Maximizes SNR at the filter output

- For Gaussian Noise, MF Detector achieves maximum detection probability for a given false alarm probability

- Also optimum for correlated Gaussian noise.

- Becomes suboptimum for non-Gaussian noise.

SIGNAL DETECTION IN ADDITIVE NOISE

Locally Optimum Detector

$$\max_{T_{LO}} \frac{dP_d}{d\theta} \quad \text{as} \quad \theta \to 0 \quad \text{(weak - signal)}$$

$$T_{LO}(X) = \sum_{i=1}^{n} s_i \frac{f'(X_i)}{f(X_i)} \qquad \text{(independent samples)}$$

- $T_{LO}(X) = T_{MF}(X)$ for Gaussian distribution
- $T_{LO}(X)$ exists for many non-Gaussian distributions
- Optimum for vanishingly small SNR and large n.
- Performance degrades for the case of

strong signals
finite

SIGNAL DETECTION IN ADDITIVE NOISE

Probability Density Functions

(1) Gaussian PDF

$$f_N(x) = \frac{e^{-\frac{x^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}, \quad \mathrm{var}(\mathrm{N_i}) = \sigma^2$$

(2) Double Exponential (DE) PDF

$$f_N(x) = \frac{e^{-\frac{|x|}{\sigma}}}{2\sigma}, \quad \mathrm{var}(\mathrm{N_i}) = 2\sigma^2$$

SIGNAL DETECTION IN ADDITIVE NOISE

(3) Contaminated Gaussian (CG) PDF

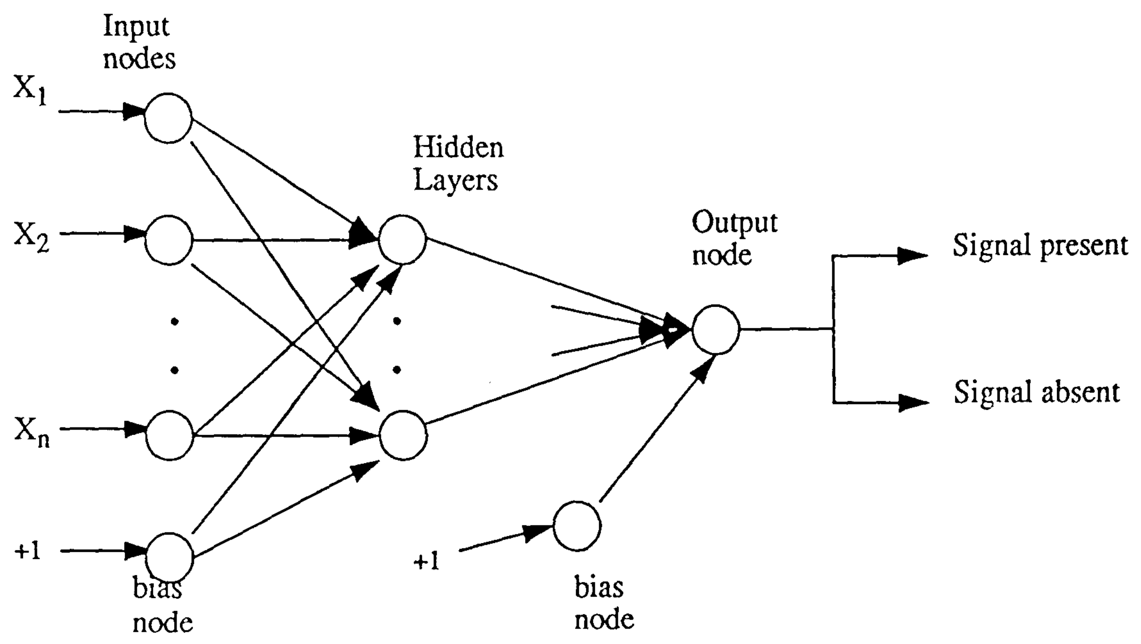
$$f_N(x) = (1-\varepsilon)\frac{e^{-\frac{x^2}{2\sigma_0{}^2}}}{\sqrt{2\pi\sigma_0{}^2}} + \varepsilon\frac{e^{-\frac{x^2}{2\sigma_1{}^2}}}{\sqrt{2\pi\sigma_1{}^2}}, \quad \sigma_0 > \sigma_1$$

$$\mathrm{Var}(\mathrm{N_i}) = (1-\varepsilon)\sigma_0^2 + \varepsilon\sigma_1^2$$

(4) Cauchy PDF

$$f_N(x) = \frac{\sigma}{[\pi(\sigma^2 + x^2)]}, \quad \mathrm{var}(\mathrm{N_i}) = \infty$$

NEURAL NETWORK FOR SIGNAL DETECTION



Block Diagram of a Neural Detector

NEURAL NETWORK FOR SIGNAL DETECTION

Training the network

Number of input nodes: n = 10

Number of hidden layers: one

Number of hidden nodes: five

Training algorithm: Back-propagation

Activation Function: Sigmoid (from 0 to 1)

During each epoch

Input	Output
noise only vector	zero
signal plus noise vector	one

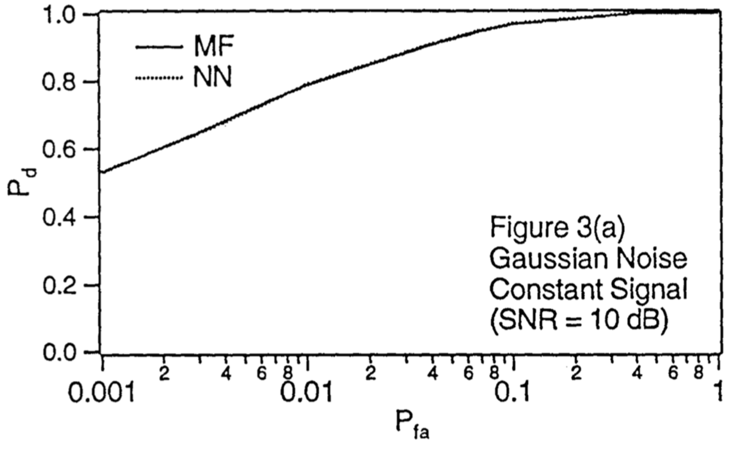
NEURAL NETWORK FOR SIGNAL DETECTION

Operation of the Neural Detector

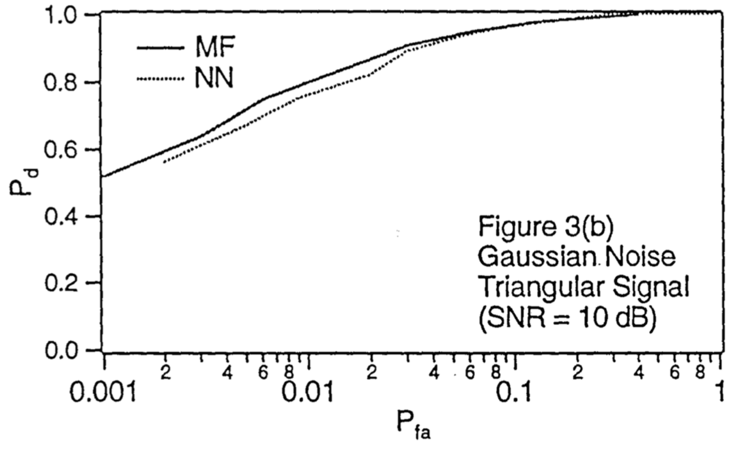
- Output node value $\geq$ 0.5 $\Rightarrow$ Signal present
- Output node value $<$ 0.5 $\Rightarrow$ Signal absent

- The trained network has some specific detection and false alarm probabilities. Network performance is optimum

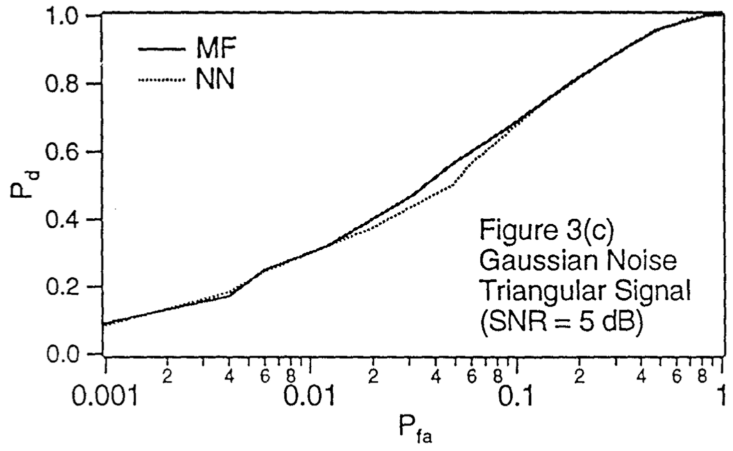
- To vary false alarm probability
  - tune the bias weight at the output node
  - alternatively, vary the output node threshold
  - Some loss of performance may be exhibited in this case



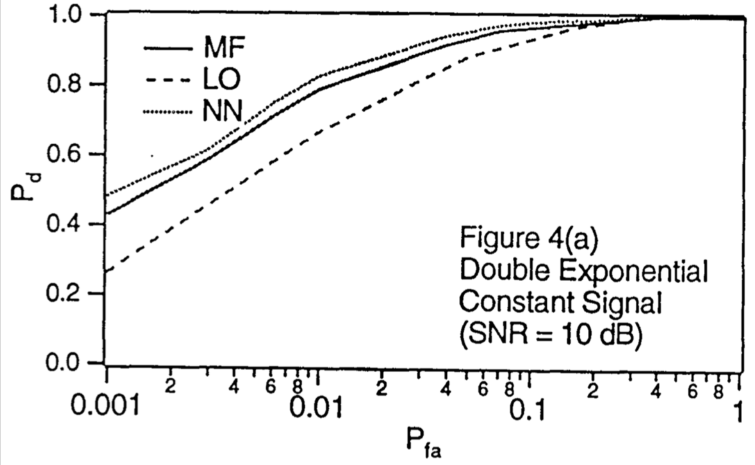
MF
NN
Figure 3(a)
Gaussian Noise
Constant Signal
(SNR = 10 dB)
P_d
P_{fa}

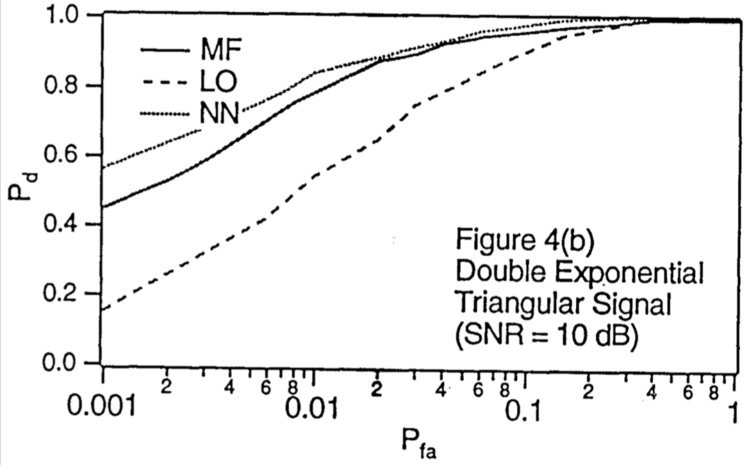


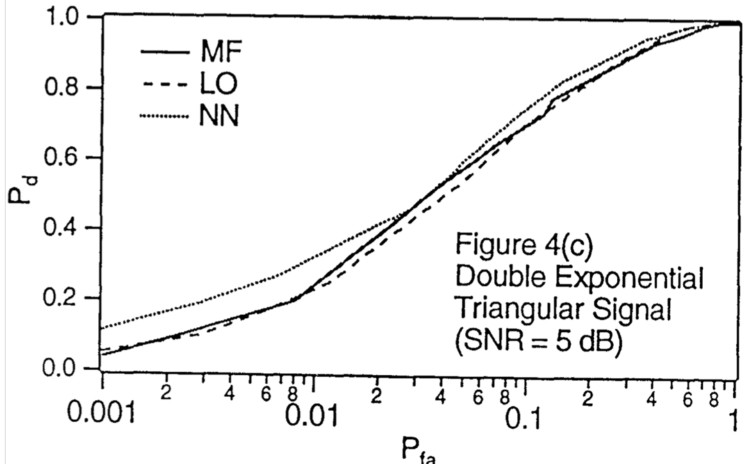
MF
NN
Figure 3(b)
Gaussian Noise
Triangular Signal
(SNR = 10 dB)
P_d
P_{fa}

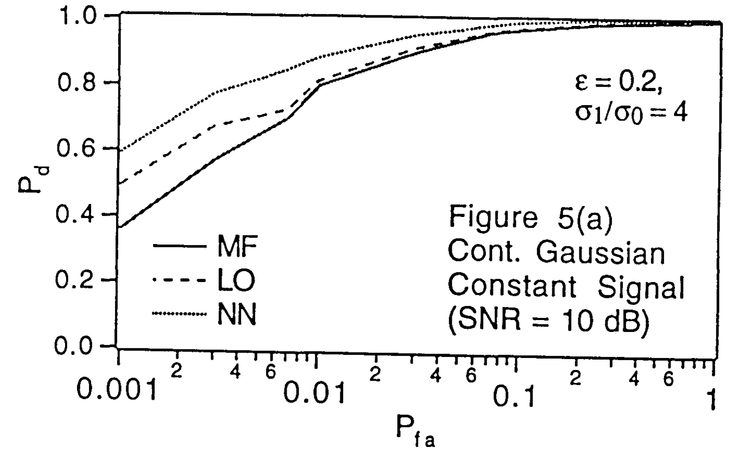


MF
NN
Figure 3(c)
Gaussian Noise
Triangular Signal
(SNR = 5 dB)
P_d
P_{fa}

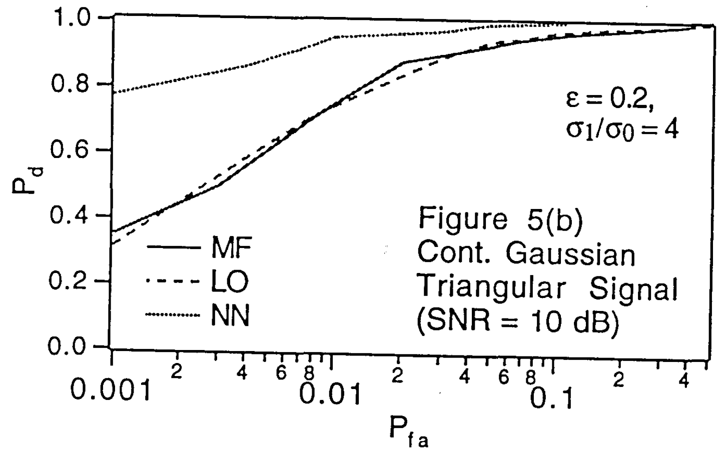

1.0
0.8
0.6
0.4
0.2
0.0
P_d
— MF
- - - LO
········ NN
Figure 4(a)
Double Exponential
Constant Signal
(SNR = 10 dB)
0.001
0.01
0.1
1
P_{fa}


1.0
0.8
0.6
0.4
0.2
0.0
P_d
— MF
- - - LO
········ NN
Figure 4(b)
Double Exponential
Triangular Signal
(SNR = 10 dB)
0.001
0.01
0.1
1
P_{fa}

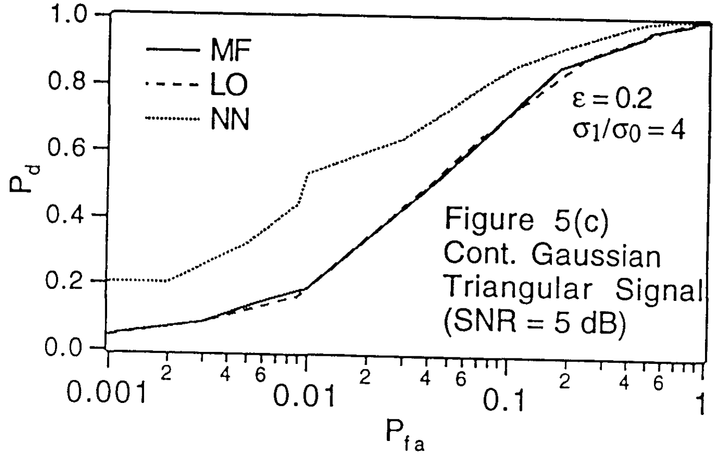

1.0
0.8
0.6
0.4
0.2
0.0
P_d
— MF
- - - LO
········ NN
Figure 4(c)
Double Exponential
Triangular Signal
(SNR = 5 dB)
0.001
0.01
0.1
1
P_{fa}



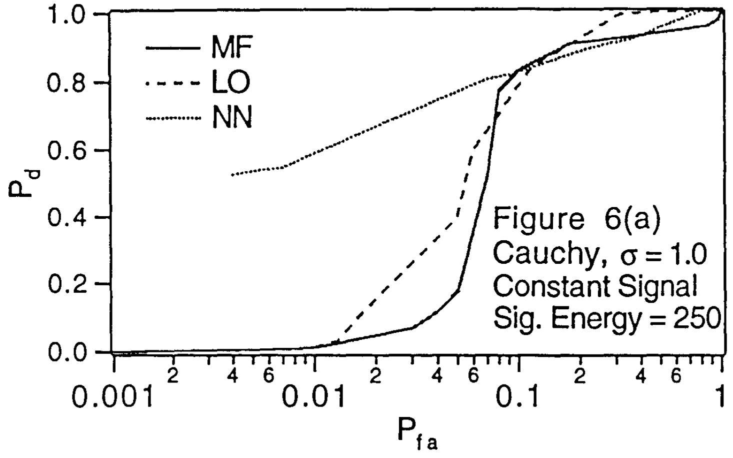
1.0
0.8
0.6
0.4
0.2
0.0
P_d
ε = 0.2,
σ1/σ0 = 4
Figure 5(a)
Cont. Gaussian
Constant Signal
(SNR = 10 dB)
MF
LO
NN
0.001
0.01
0.1
1
P_fa



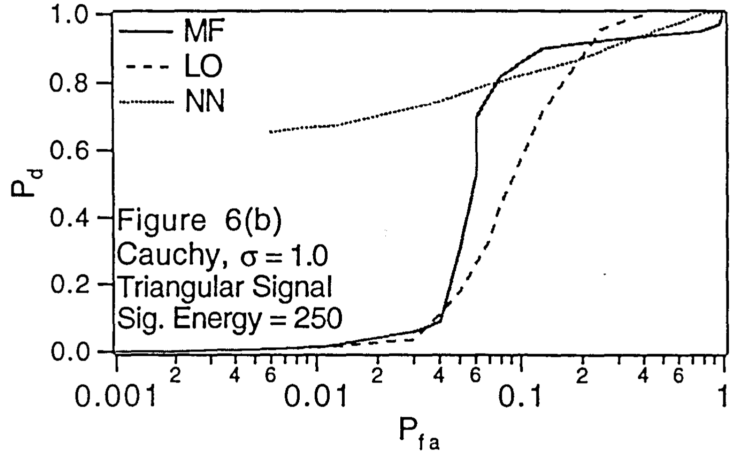
1.0
0.8
0.6
0.4
0.2
0.0
P_d
ε = 0.2,
σ1/σ0 = 4
Figure 5(b)
Cont. Gaussian
Triangular Signal
(SNR = 10 dB)
MF
LO
NN
0.001
0.01
0.1
P_fa



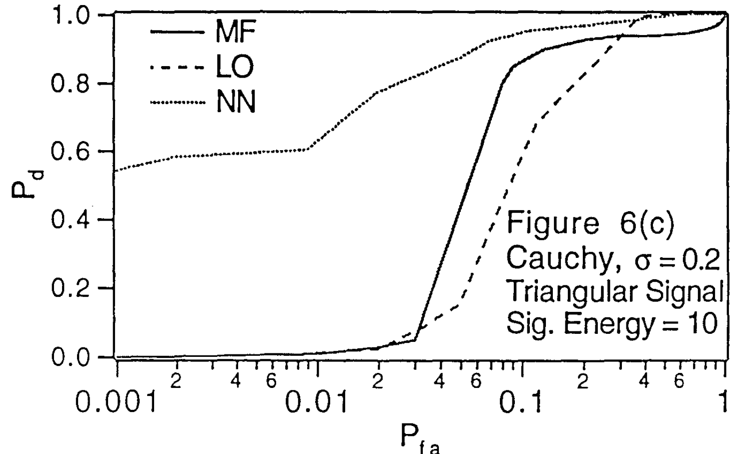
1.0
0.8
0.6
0.4
0.2
0.0
P_d
MF
LO
NN
ε = 0.2
σ1/σ0 = 4
Figure 5(c)
Cont. Gaussian
Triangular Signal
(SNR = 5 dB)
0.001
0.01
0.1
1
P_fa



MF
LO
NN
Figure 6(a)
Cauchy, σ = 1.0
Constant Signal
Sig. Energy = 250
P_d
P_{fa}



MF
LO
NN
Figure 6(b)
Cauchy, σ = 1.0
Triangular Signal
Sig. Energy = 250
P_d
P_{fa}



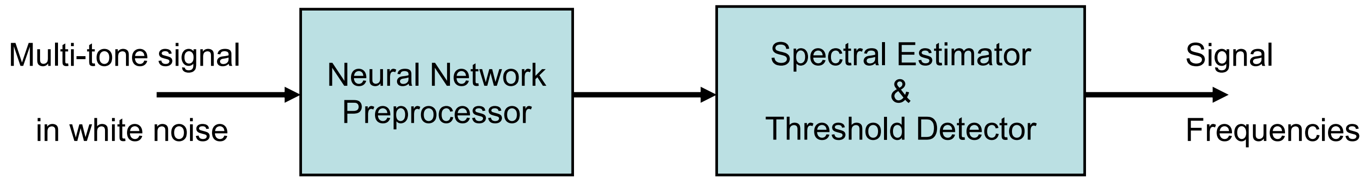
MF
LO
NN
Figure 6(c)
Cauchy, σ = 0.2
Triangular Signal
Sig. Energy = 10
P_d
P_{fa}

The Multi-tone Detection and Estimation

- Neural Networks for Noise Reduction
- Neural Network Filters
- Parallel Bank of NN Filters
- Tunable NN Filters
- Simulation Results
- MFSK Receiver Application

The Multi-tone Detection and Estimation Problem

Given a multi-tone signal corrupted by additive white noise, finding the approximate frequency band in which it lies and estimating the frequencies using spectral estimators



Multi-tone signal
in white noise
Neural Network Preprocessor
Spectral Estimator & Threshold Detector
Signal
Frequencies

Neural Networks for Noise Reduction

Non-linear mapping from the noisy signals to the noiseless ones obtained by the non (semi) linear sigmoidal squashing function. Bandpass filtering removes the noise outside the band and improves the signal to noise ratio.

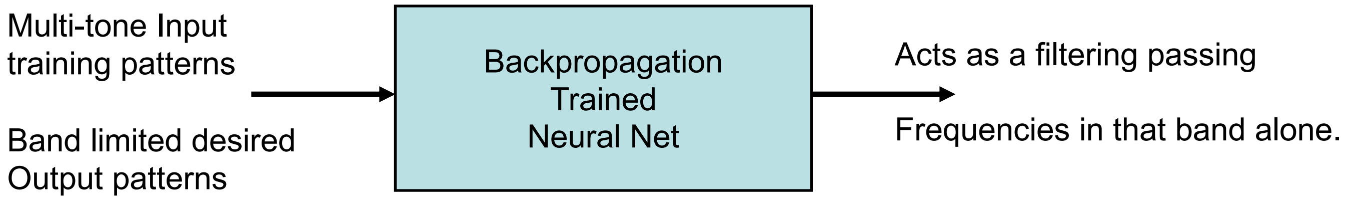
Ideally, we expect the network to learn to separate the signal from the noisy signal for the entire range of frequencies, in spite of the number of sinusoids in the signal. In practice, it detects well only when a single sinusoid is present. For more than one sinusoid, the net learns only one dominant sinusoid.

The Multi-tone Detection and Estimation Problem

This very limitation leads to a solution.

Neural Networks as filters

When trained in a specific frequency band using the standard backpropagation algorithm, the Neural Network acts as a good filter giving:
High gain in the passband, good attenuation in the stopband & sharp transitions



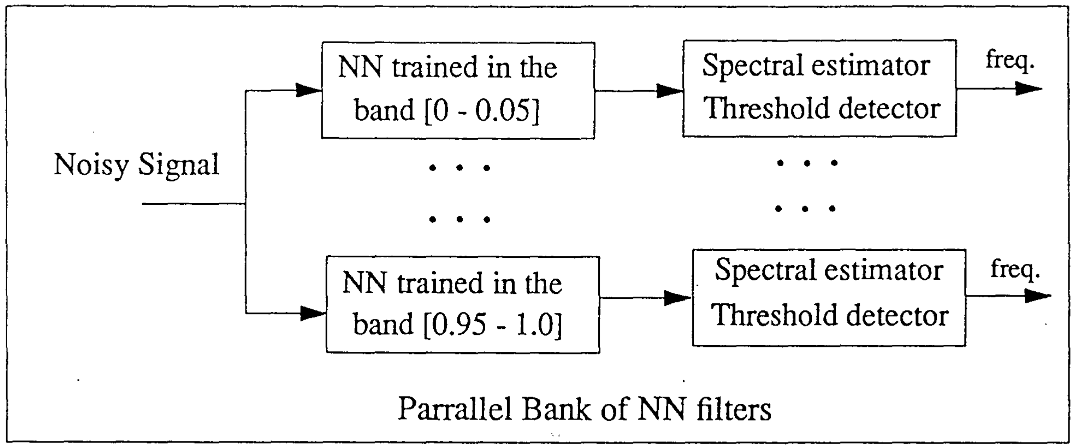
Problem Solution:

Partition the normalized frequency spectrum into 20 bands.
Train a bank of 20 such Neural Nets in each of these bands.

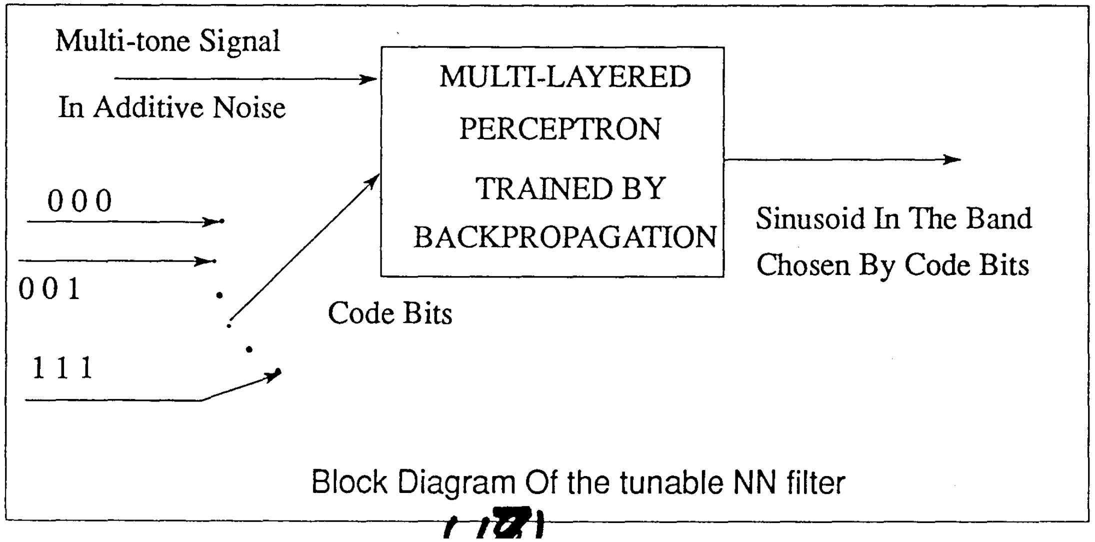
Drawbacks:

Even the noise in the passband gets amplified & leads to High false alarm rates
The stopband signals get passed in some cases. (Poor reliability on detection)

Tunable NN Filter


Noisy Signal
NN trained in the band [0 - 0.05]
Spectral estimator Threshold detector
freq.
NN trained in the band [0.95 - 1.0]
Spectral estimator Threshold detector
freq.

Parrallel Bank of NN filters



Block Diagram Of the tunable NN filter

S.S.Rao & S.Sethuraman, "A Tunable Neural Network Filter for Multi-tone Detection", (Published in the MILCOM'92 Conference records, San Diego, CA)

Tunable Filter Performance & Applications

- High gain the band chosen by the code (Selection)
- Large Attenuation outside the chosen band (Rejection)
- An average of (<)10% probability of false alarm at 0 dB.
- When trained at a particular SNR, there is a graceful degradation with higher SNRs
- Good generalization from the training data.
- Parsimony in weights compared to the parallel bank of NN Filters.

How is detection done ?

Place the noisy signal at the input. Sweep through 8 codes.

Square and sum the output and threshold it. If the threshold is exceeded, then it is declared as 'signal present', in the band corresponding to the code.

Where can this be used ?

MFSK, where $M = 2^k$ frequencies are used to represent the combinations of k bits. In these cases, the frequencies of interest are know beforehand and can be used for training

Hopfield Neural Network for Spectral Estimation

The model for an AR process is given by

$$x(n) = \sum_{i=1}^{P} a_i x(n-i) + e(n)$$

The AR parameters can be obtained by minimizing

$$E\left[\left\{x(n) - \sum_{i=1}^{P} a_i x(n-i)\right\}^2\right]$$

Replacing the ensemble average by time average,

$$\sum_{n=P+1}^{L} \left\{x(n) - \sum_{i=1}^{P} a_i x(n-i)\right\}^2 = \sum_{n=P+1}^{L} \left\{x(n) - a^t x_n\right\}^2$$

where,

$$\mathrm{x_n} = [\mathrm{x(n-1)\ x(n-2)\ \ \ldots\ldots..\ \ x(n-P)}]^{\mathrm{t}}$$

$$\mathrm{a} = [\mathrm{a_1\ a_2\ \ldots\ldots..\ \ a_p}]^{\mathrm{t}}$$

Hopfield Neural Network for Spectral Estimation

- Defining

$$y = [x(n)\ \ x(n-1)\ \ x(n-2)\ \ \ldots\ldots\ldots\ \ x(L)]^t$$

$$X = [x_n\ \ x_{n+1}\ \ldots\ldots\ldots\ \ x_L]^t$$

$$J = \left\|(y - Xa)\right\|^2 = [(y - Xa)^t (y - Xa)] = y^t y + a^t X^t X a - a^t X^t y - y^t X a$$

Since $y^t y$ is independent of a, minimizing J is equivalent to minimizing,

$$J = y^t y + a^t X^t X a - a^t X^t y - y^t X a = \sum_{i=1}^{P} \sum_{j=1}^{P} X_i X_j^t a_i a_j - 2 \sum_{i=1}^{P} y^t X_i a_i$$

Comparing this with the Lyapunov energy function,

$$E = -\frac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{P} W_{ij} v_i v_j - \sum_{i=1}^{P} I_i v_i$$

$$W_{ij} = -X_i^t X_j, \text{ and } I_i = y^t X_i$$

Now, on convergence the output of the network will give the vector a.

Hopfield Neural Network for Spectral Estimation

- The activation function chosen for the neurons is a soft limiter given by

$$
\begin{aligned}
a_i &= net_i / b && \text{if } |\, net_i \,| < b\beta \\
&= b && \text{if } |\, net_i \,| > b\beta \\
&= -b && \text{if } |\, net_i \,| < -b\beta
\end{aligned}
$$

where

$$net_i(t+1) = net_i(t) + \sum W_{ij} a_i + I_i$$

This leads to an iterative method of computing the AR coefficients of a given time series

Hopfield Neural Network for Spectral Estimation

- Given now the AR coefficients, the power spectral density can be written as,

$$P(w) = \frac{T\sigma^2}{\left|1 - \sum_{i=1}^{P} a_i e^{-jw_i T}\right|^2}$$

where T : sampling time
σ^2 : variance of the white noise

- For the complex sinusoids case, the weights and inputs to the network are modified as follows:

$$W_{ij} = -\text{Real}(X_i^H X_j)$$

$$I_i = \text{Real}(y^H X_i)$$

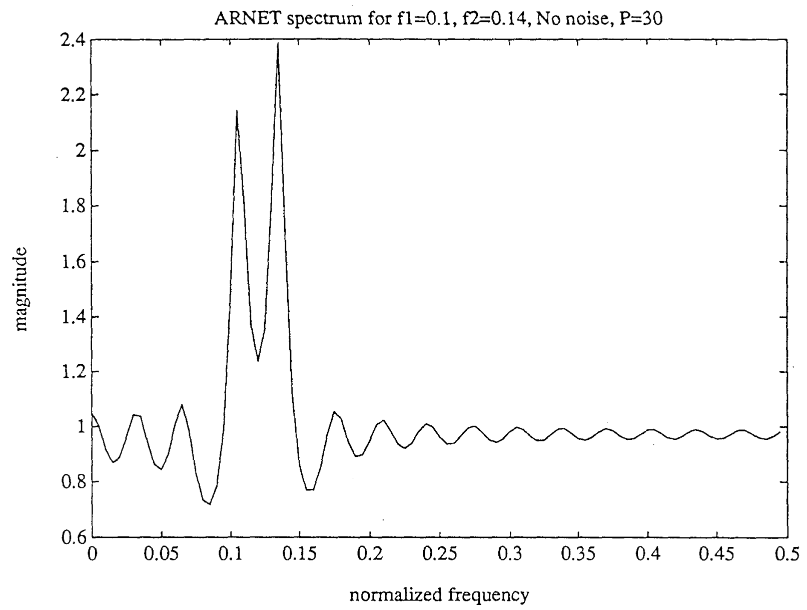
Hopfield Neural Network for Spectral Estimation

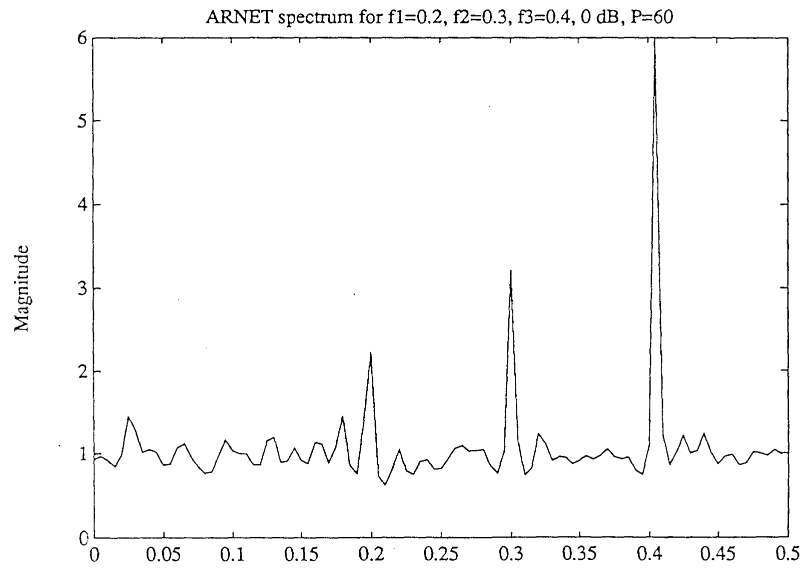
- Hopfield Neural Network for spectral estimation is found to be

  robust upto 3 dB

  poor performance for closely spaced sinusoids

  no spurious peaks for large model orders


ARNET spectrum for f1=0.1, f2=0.14, No noise, P=30
magnitude
normalized frequency
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


ARNET spectrum for f1=0.2, f2=0.3, f3=0.4, 0 dB, P=60
Magnitude
6
5
4
3
2
1
0
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5

Unsupervised Learning

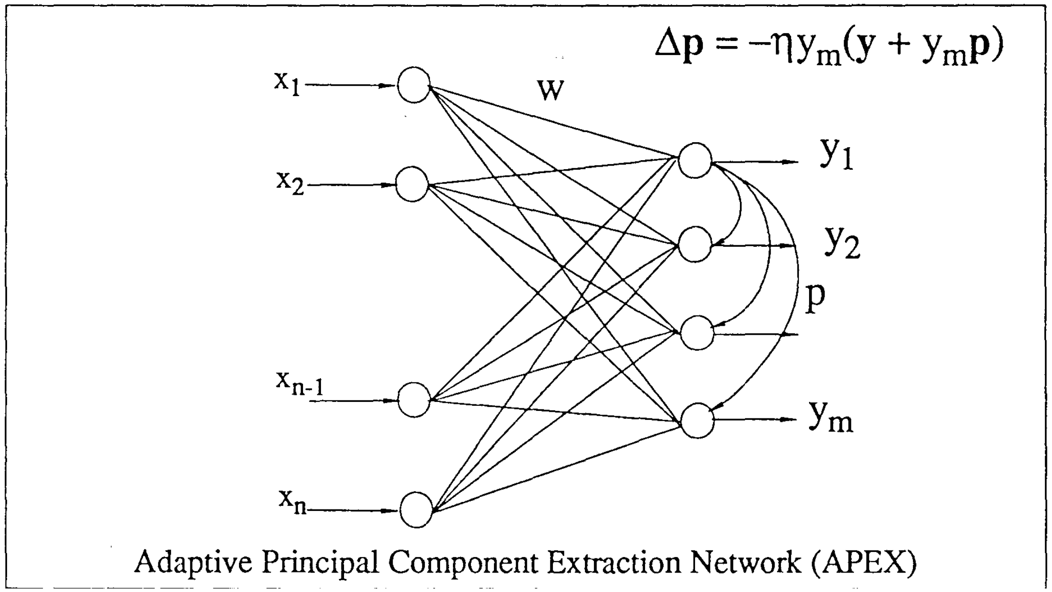
Hebbian Rule: $\Delta w_j = \eta y x_j$, $y = \mathbf{x}^t\mathbf{w}$ (unbounded)

Oja's modified Rule: $\Delta\mathbf{w} = \eta y(\mathbf{x}-y\mathbf{w}) = \eta(\mathbf{I}-\mathbf{w}\mathbf{w}^t)\,\mathbf{x}\mathbf{x}^t\mathbf{w}$ (normalized)

Oja's Rule for multiple output case: $\Delta\mathbf{w}_{ij} = \eta y_i(x_j - \Sigma y_k w_{kj})$

Oja's rule is used to extract the principal eigenvectors of $\mathbf{C} = E(\mathbf{x}\mathbf{x}^t)$

Other modifications – with orthogonalizing lateral connections as in APEX



Adaptive Principal Component Extraction Network (APEX)

E. Oja, "A simplified Neuron model as a Principal Compnent Analyzer", J.Math.Biology, Vol.15, pp.267-273, 1982
S. Y .Kung & K. I. Diamantaras, "A neural network learning algorithm for adaptive principal component extraction:, pp.861-864, Proc. Of ICASSP'90

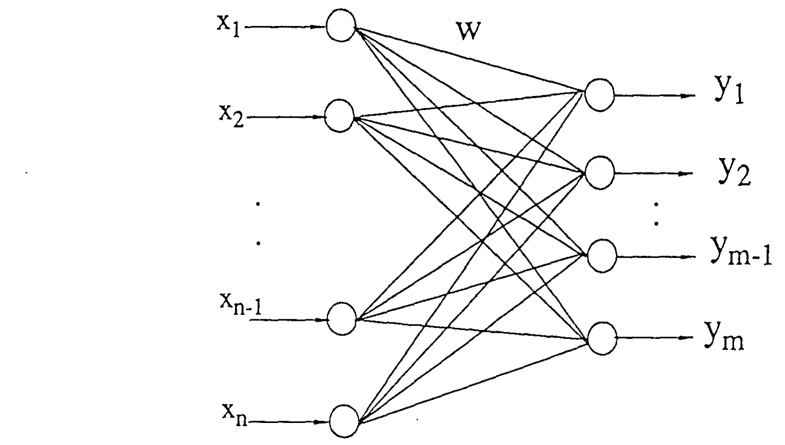


Fig.5.a :Single layer feedforward network for principal components extraction

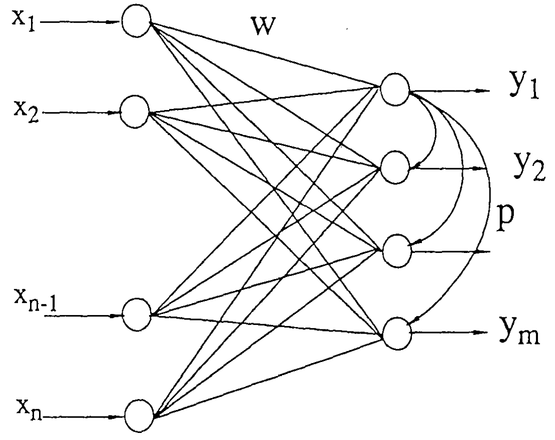
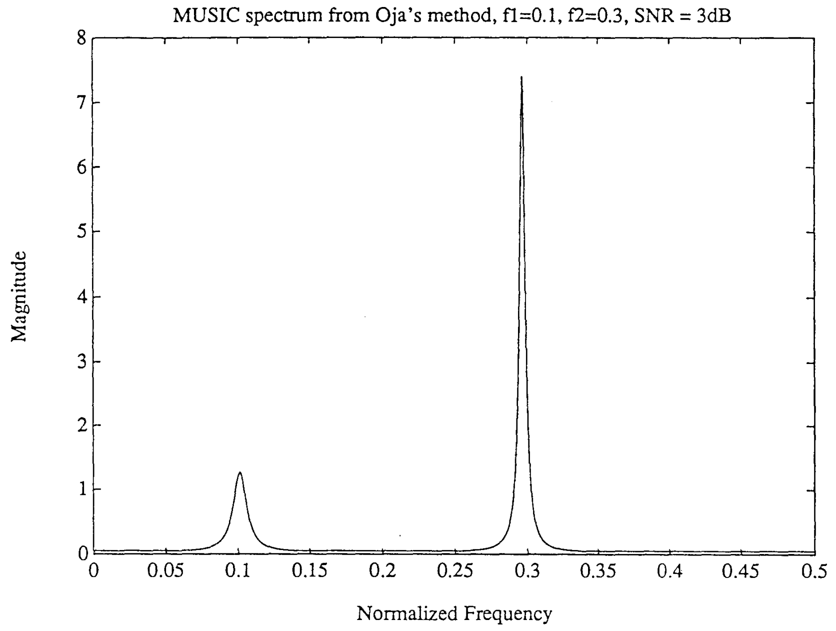
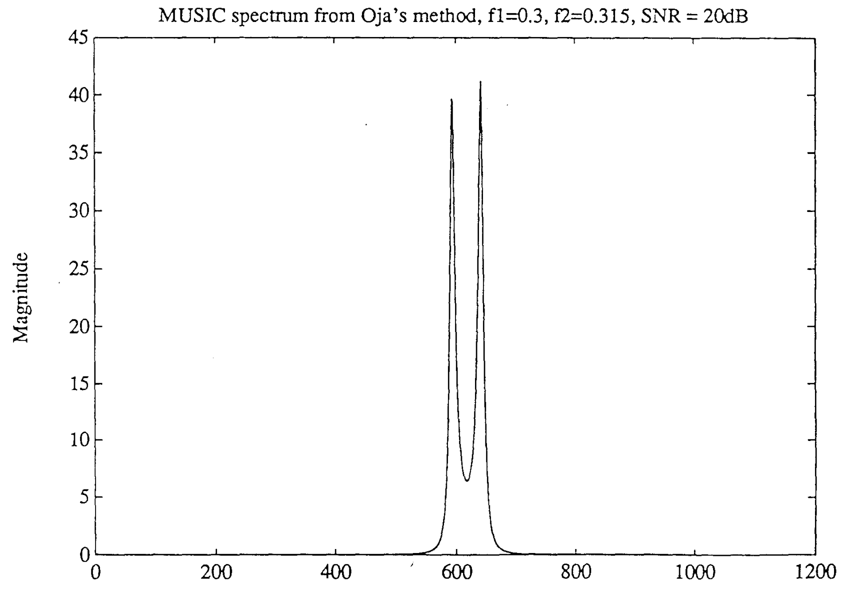


Fig.5.b: Adaptive Principal Component Extraction Network (APEX)


MUSIC spectrum from Oja's method, f1=0.1, f2=0.3, SNR = 3dB
Magnitude
Normalized Frequency
0
1
2
3
4
5
6
7
8
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5


MUSIC spectrum from Oja's method, f1=0.3, f2=0.315, SNR = 20dB
Magnitude
0
5
10
15
20
25
30
35
40
45
0
200
400
600
800
1000
1200

NEURAL NETWORKS: The Representation by Approximation Schemes

Approximation Problem:

Let f(x) be a real-valued function defined on a set X, and let F(A,x) be a real-valued approximating function depending continuously on x ε X and on n parameters, A. Given the distance function d, determine the parameters A* ε A such that

$$\boldsymbol{d}\,[\,\boldsymbol{F}(\boldsymbol{A}^*, x), f(x)\,] \leq \boldsymbol{d}\,[\,\boldsymbol{F}(\boldsymbol{A}, x), f(x)\,]$$

for all A* ε A

A is the space in which parameters lie and is usually the ordinary Euclidean space. The distance d is a "measure of the approximation" and is generally given as the L^p norm of the difference **F(A*,**) – f(x); i.e.
d = L^p[**F(A***, x) – f(x)]

$$d = \left[\int_0^1 \left| F(A^*, x) - f(x) \right|^p dx \right]^{\frac{1}{p}} \qquad p \geq 1$$

The Representation by Approximation Schemes

The solution, the approximation problem is said to be a best approximation of the underlying function

- For example, a linear approximation is given by

  F(W,X) = WX

where	W	:	a m x n matrix of coefficients
	X	:	n x 1 vector of input variables
	Network	:	n inputs, m outputs and no hidden nodes

- For example, spline fitting, single layer BPBs etc. can be represented by

  F(W,X) = W Φ (X)

  as a linear combination of a suitable set of basis functions

  This corresponds to a network with one hidden layer.

- For example, Backpropagation networks with multiple layers may be expressed as

$$F(W,X) = \sigma\left(\sum_n W_n \sigma\left(\sum_i v_i \sigma\left(........\sum_j u_j X_j\right)\right)\right)$$

where σ is the sigmoidal function and W_n, v_i, u_j …. are the adjustable coefficients

The Representation by Approximation Schemes

- Approximation by Expansion on an Orthonormal Basis

Consider a function F(X) which is assumed to be continuous in the range [0,1]. Let $\Phi_i(x)$, i = 1, 2, …∞ be an orthonormal set of continuous function in [0,1]. Then, F(X) possesses a unique L2 approximation (Rice, 1964) of the form:

$$F(C, X) = \sum_{k=1}^{n} C_k \Phi_k(X) \qquad \text{........ (1)}$$

where $C = [C_1, C_2, \ldots. C_n]^T$ are given by the projection of F(X) onto each basis function, i.e.

$$C_k = \int_0^1 F(X)\Phi_k(X)dX \qquad \text{......... (2)}$$

The set $[\Phi_k(X)]$ is complete as we include more basis functions and in the limit, error tends to zero.

The Representation by Approximation Schemes

If we include K terms in the approximation, the error is given by

$$e_k^2 = \int_0^1 \left[F(X) - \sum_{k=1}^{K} C_k \Phi_k(X) \right]^2 dX \qquad \text{.........} \quad (3)$$

The larger the value of the coefficient, C_k, the greater the contribution of the corresponding basis function, $\Phi_k(X)$ in the approximating function. This then provides a criterion for picking the most important activation function in each hidden unit of the network.

- Smallest Network by Approximation theory:

Given an orthogonal set of functions, $\Phi_k(X)$, $k = 1, 2, \ldots. \infty$, the smallest network for approximating a function F(X) that is continuous in [0,1], with a desired accuracy is achieved by selecting basis functions corresponding to the largest coefficients C_k, calculated by (2). The resulting error is defined by L^2 error given by (3)

The Representation by Approximation Schemes

- Wavelets as Basis Functions for Neural Networks

A family of wavelets is derived from translations and dilations of a single function. If $\Psi(X)$ is the starting function, to be called Wavelet, the members of the family are given by

$$\frac{1}{\sqrt{s}}\Psi\left(\frac{X-u}{s}\right) \quad \text{for} \quad (s,u)\in R^2$$

s: indicating dilation

u: indicating translation

The Representation by Approximation Schemes

If the input, X is defined in a discrete domain and if the dilation of the wavelet is always by the factor of 2, the resulting family of discrete dyadic wavelets is represented by

$$\sqrt{2^{-m}}\Psi(2^{-m}X\text{-}k) \quad \text{for} \quad (m,k) \in Z^2$$

m: The size of the dilation (as a multiple of 2)

k: The discrete-step translation of the wavelet, $\Psi(X)$

They are related to the continuous parameters

$$s = 2^m, \qquad u = k2^m$$

The Representation by Approximation Schemes

Orthonormal wavelets (Mayer, 1985; Daubechies, 1988; Mallet 1989; Strag, 1989)

Examples are: Meyer wavelet, the Haar wavelet, the Battle – Lemarie wavelets, Daubechies compactly supported wavelets.

Reference Texts:

Wavelets: Volume I, II, Charles K. Chui, *Academic press, INC* 1992

The Representation by Approximation Schemes

- Wave-Nets: (B.R. Bakshi and G. Stephanopoulous 1992)

  A wave-net consists of input and output nodes and two types of hidden layer nodes: wavelet nodes of Ψ–modes, and scaling function nodes, or Φ–nodes.

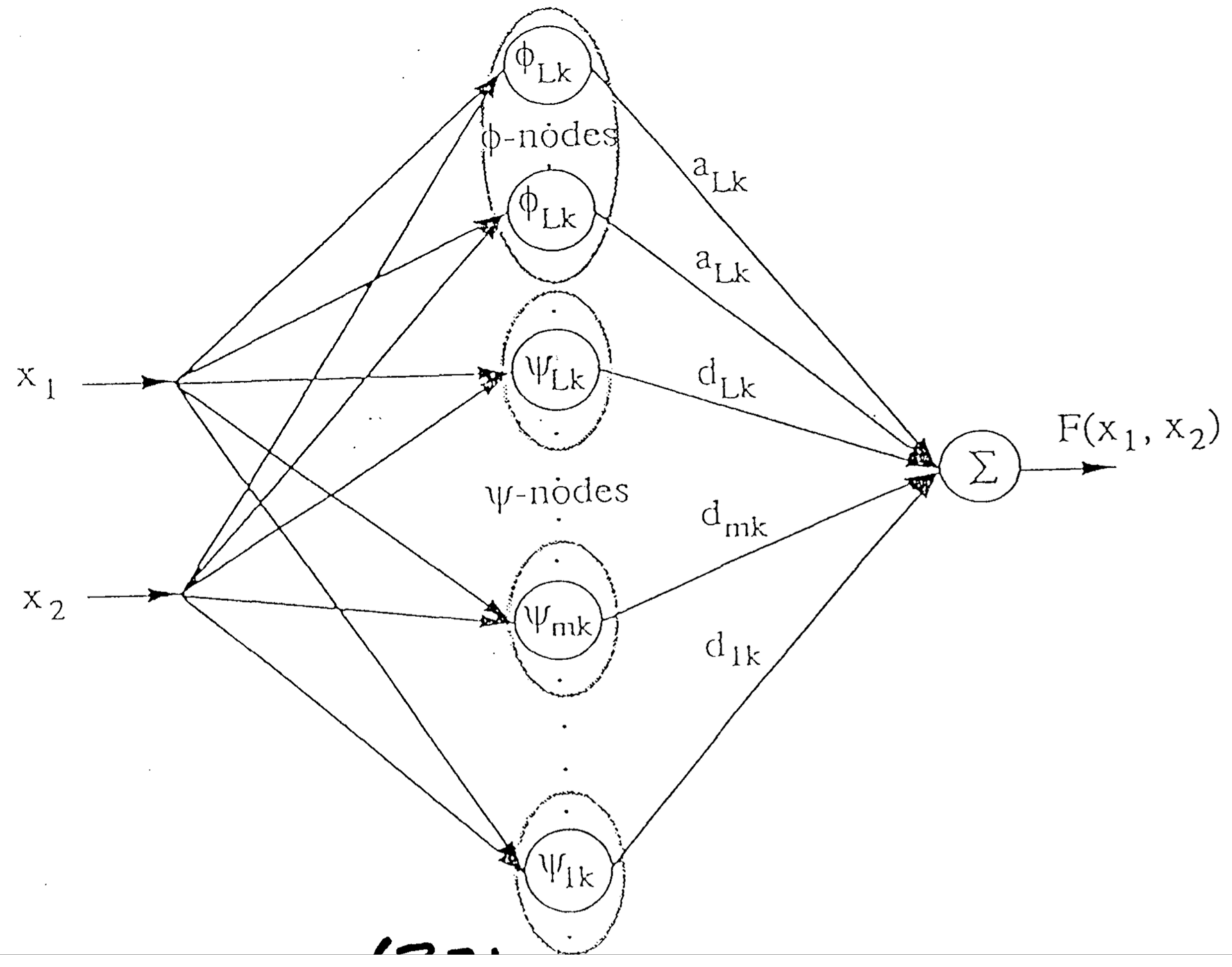
The Representation by Approximation Schemes

The basis functions associated with the hidden layer nodes are:

a. Basis functions for Φ-nodes: $\Phi_{LK}(X)$, $K = 1, 2, \ldots., n_L$; the translates of the dilated scaling function, $\Phi(X)$, which form the orthonormal basis for the approximation of the unknown function, $F(X)$, at the L-th coarsest resolution.

b. Basis functions for Ψ-nodes: $\Psi_{mk}(X)$, $m = 1, 2, \ldots., L$, and $k = 1, 2, \ldots., n_m$; the translates of the dilated wavelet, $\Psi(X)$, which form the orthonormal basis for detail of function, $F(X)$, at each resolution.

Wave - Nets


ϕ_{Lk}
ϕ-nodes
ϕ_{Lk}
a_{Lk}
a_{Lk}
x_1
ψ_{Lk}
d_{Lk}
$F(x_1, x_2)$
Σ
ψ-nodes
d_{mk}
x_2
ψ_{mk}
d_{1k}
ψ_{1k}

Wave - Nets

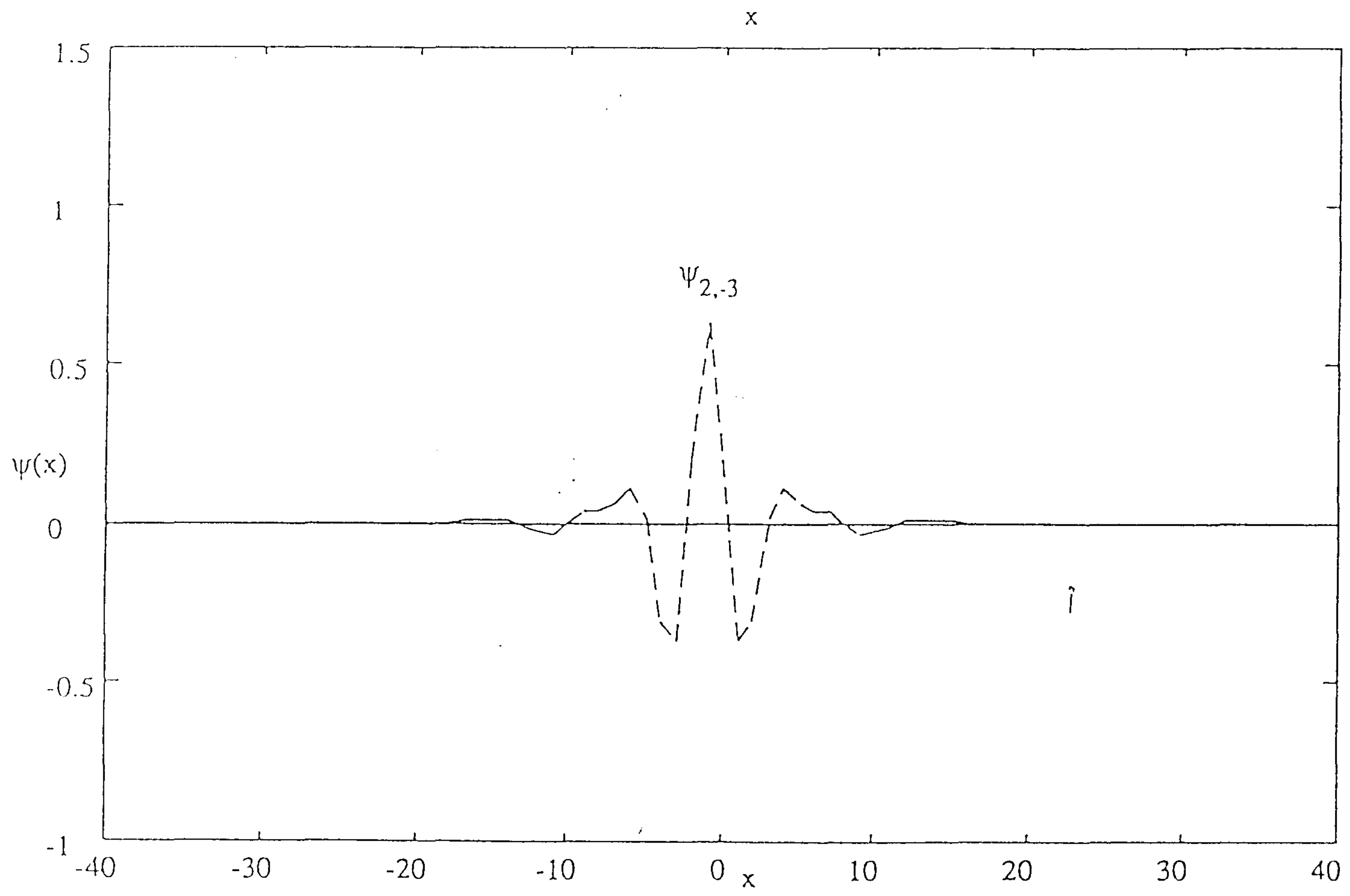
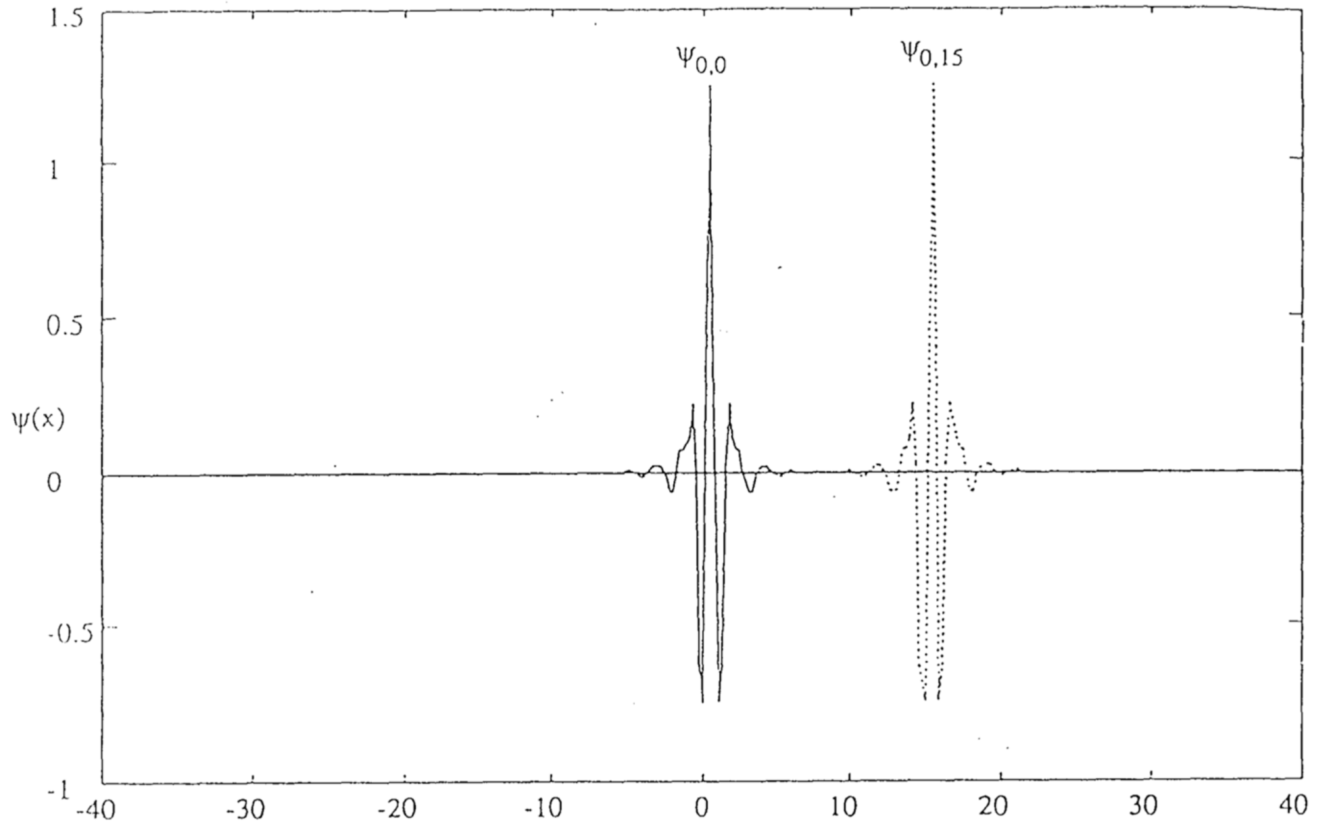
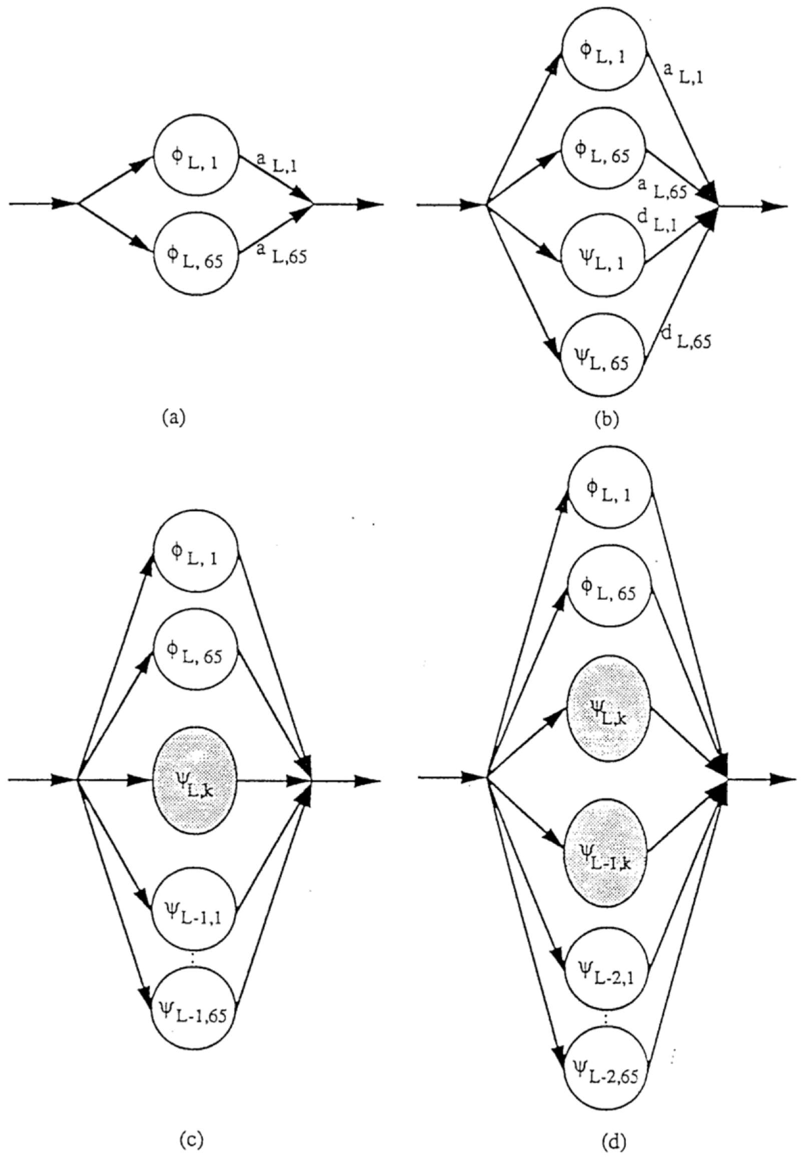


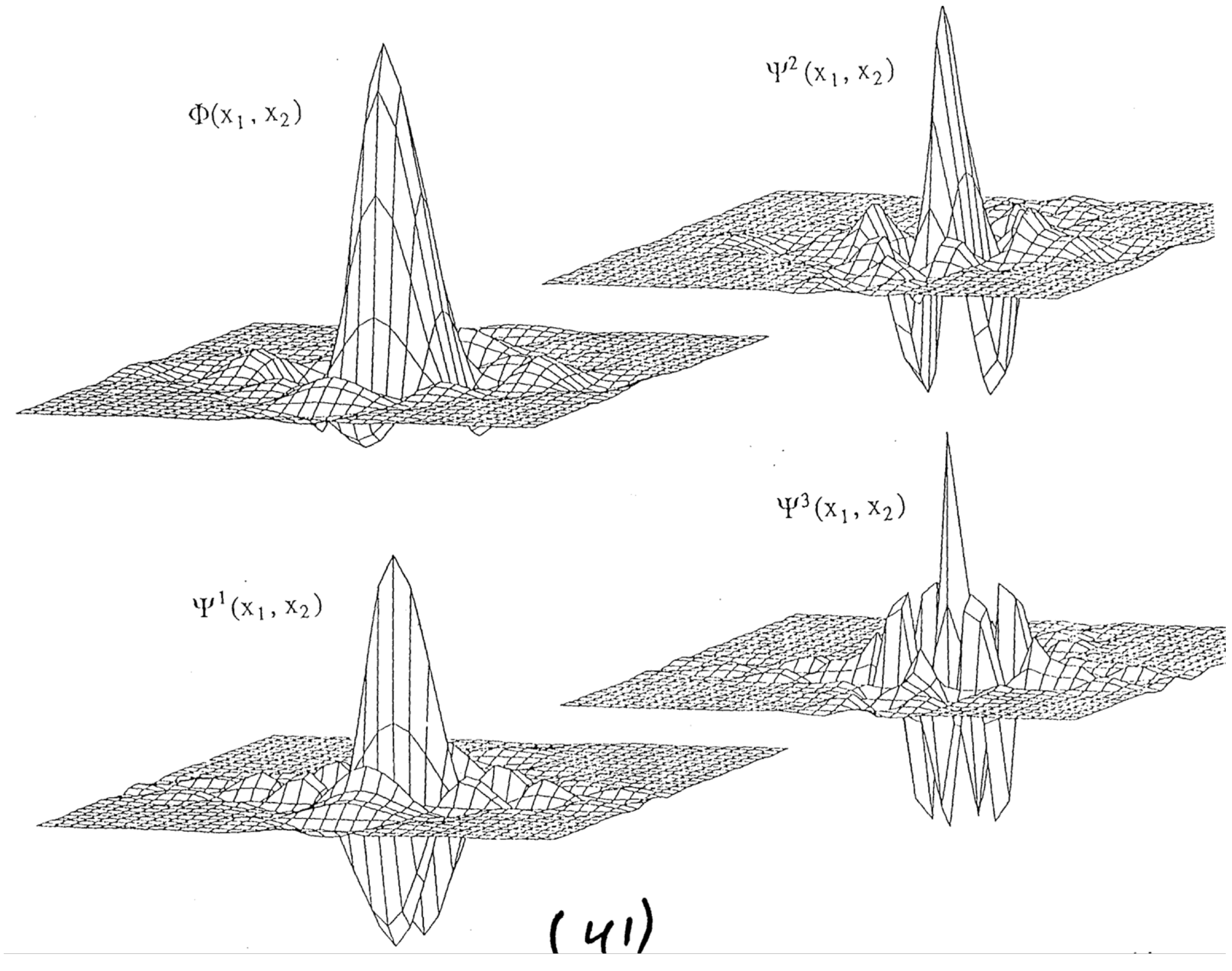
Figure 2. Translation and dilation of Battle-Lemarie wavelet

Wave - Nets





Construction of a *Wave-Net* at different resolutions.
(a) ϕ-nodes at scale L, (b) with ψ-nodes at scale L,
(c) with ψ-nodes at scale (L-1), (d) with ψ-nodes at scale (L-2)



(41)