A FUNCTIONAL LINK NETWORK USING ORDERED BASIS FUNCTIONS

by

SAURABH SUREKA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2007

ACKNOWLEDGEMENTS

I express my heartfelt gratitude to Dr. Michael Manry for his guidance and support during the thesis work. He saw the work from inception to fruition and provided all the help to make this work possible. I admire his subject expertise, contribution and devotion to the field of Neural Networks, and incessant help to his students in various forms viz. teaching, regular laboratory visits, immediate feedback and motivation to better understand the field of Neural Networks.

The EE coursework at UTA and my undergraduate institute SSGMCE in India provided the fillip and toolkit to weave the pieces of this work together. Hence a sincere thanks to all the teachers who selflessly strive to spread education to whom I dedicate this work. Thanks to Dr. Johnathan Bredow and Dr. Stephen Gibbs and for a patient review of the thesis work and thanks also to Dr. Gibbs for giving me a small opportunity to learn and teach at UTA.

A special mention of my various colleagues and supporters at GTL, Infosys, IPNNL, Qualcomm and online research community, who have either directly or indirectly but certainly contributed to this experience. A big thanks to my family and friends for being there.

Finally given the time-dependent boundary conditions this appears an eloquent pit stop of the unbounded learning journey.

March 15, 2007

ABSTRACT

A FUNCTIONAL LINK NETWORK USING ORDERED BASIS FUNCTIONS

Publication No. _____

Saurabh Sureka, M.S.

The University of Texas at Arlington, 2007

Supervising Professor: Michael T. Manry

A new function approximation and classification network based on Functional Link Network (FLN) with orthonormal Polynomial Basis Functions (PBF) is presented. By using an iterative Gram-Schmidt procedure, the PBF's are orthonormalized, ordered and selected based on their contribution to minimize the Mean Square Error (MSE). Linearly dependent and less useful PBF are detected and eliminated at an early stage thereby improving the approximation capabilities and reducing the possibility of combinatorial explosion. The number of passes through the data during network training is minimized through the use of correlations. A one-pass method is used for validation and network sizing. Equivalent function approximation and classification networks are designed and simulation examples are presented. Results for the Ordered

FLN are compared with those for the FLN, Group Method of Data Handling (GMDH), and Multi-Layer Perceptron (MLP), Nearest Neighbor Classifier (NNC) and Piecewise Linear Classifier (PLNC).

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The field of Neurocomputing and Polynomial Networks has developed rapidly in the last few decades. Pioneering work for Neurocomputing dates back to Warren McCulloch and Walter Pitts work on modeling neuron activity with electrical circuits in 1943[1]. One approach for learning is the supervised learning technique in which the training data comprises both the input and desired output patterns. The Group Method of Data Handling (GMDH) is a self-organized, supervised learning type Polynomial Network developed by cyberneticist Dr A.G. Ivakhnenko in 1966[2][3]. Concurrent research primarily in US, Europe and Japan along with advances in semiconductor engineering has effectively widened the spectrum of end-applications for the Polynomial Networks and GMDH.

Polynomial Networks (PN) and GMDH are also sometimes referred as Polynomial Neural Networks (PNN) in the research community. Together they provide novel solutions to a set of problems including function learning, optimization, interpolation, structure identification, classification and associative learning. These problems are commonly encountered in the fields of Adaptive Control, Optimization and Scheduling, Signal Processing, Pattern Recognition, Data Mining, Artificial Intelligence and Computer Vision. Successful implementations of a few end-applications include face and handwriting recognition, natural language processing,

image compression using auto-associativeness, remote sensing, servo control mechanisms used in interplanetary probes and expeditions and short and long-term forecasting of financial market, weather, earthquakes and temperature[4-8].

1.1 History and Research Trends in Polynomial Networks

Polynomial Networks as well as most other networks like the Multi-Layer Perceptron (MLP), Radial Basis Function (RBF) network, Piecewise Linear Network (PLN) derive their universal approximation capabilities from the Weierstrass approximation theorem (1885), expanded upon by Stone (1948)[9]. The growth in Neural Networks after the initial excitement from 1944 to 1969 met a sudden death due to lack of funding, computational limitations and the often cited work of Minsky and Papert[10]. But some work by the likes of Werbos[11], Fukushima[12], Hopfield[13], Kohonen[14], Grossberg[15] and many others provided a motivation for a new direction in the field. The evolution of Polynomial Networks however has a steady but slow growth from the times of Ivakhnenkho who developed the GMDH network in 1966[2-3]. Variations of GMDH-type networks and learning algorithms have been proposed since then. Pao in his pioneering work introduced a Functional Link Network (FLN)[16] which like the GMDH tries to approximate a function with a Kolmogorov-Gabor polynomial[17]. Polynomial networks have been shown to have universal approximation capabilities using few basis functions like Trigonometric functions, algebraic polynomials, orthogonal polynomials, Chebyshev polynomials, Hermite polynomials, Legendre polynomials, splines, ridged polynomials. The practical

implementations of these are limited by an exponential increase in the required number of basis functions with an increase in the network inputs and / or degree of approximation. To address this problem of combinatorial explosion researchers have used Sigma-Pi Networks, Genetic Algorithms and other evolutionary approaches[18-19]. However, faster learning algorithms, methods to generate and handle higher order polynomials, achieving optimal network design and size are topics of great interests and open research.

1.2 OFLN Scope

This thesis introduces an application of a fast iterative Gram-Schmidt Orthonormalization procedure to a Functional Link Network (FLN) and the resultant network obtained is called the Ordered Functional Link Network (OFLN). The functional approximation capabilities of OFLN are theoretically justified by the Weierstrass approximation theorem that states that polynomial approximation gets arbitrarily close to any continuous function as the polynomial order is increased[9].

1.3 Thesis Organization

A brief introduction to the structure of function approximation and review of a few polynomial networks is presented in Chapter II. An Iterative Gram-Schmidt Orthonormalization procedure, structural representation and implementation of the OFLN are presented in Chapter III. The approximation capabilities of OFLN and iterative Output-Reset (OR) algorithm that decreases the classification error form a good combination of OFLN as a regression-type classifier and are discussed in Chapter

IV. Results from comparison of few benchmark examples of function approximation and classification are compared with other networks in Chapter V. Conclusions and Future Work constitute Chapter VI.

CHAPTER 2

REVIEW OF POLYNOMIAL NETWORKS

2.1 Structure of Function Approximators

The problem of function approximation can be stated as determination of the closest functional relationship that maps the N-dimensional input vector space to M-dimensional output vector space. Consider a system with an input $\mathbf{x}$ ($\mathbf{x} \in \Re^N$) with probability density function $f_{\mathbf{x}}(\mathbf{x})$ and the desired output $\mathbf{y}$ ($\mathbf{y} \in \Re^M$) which is based on the aposteriori conditional density $f_{\mathbf{y}}(\mathbf{y}|\mathbf{x})$. Then, the joint density of independent and identically distributed observations $f_{\mathbf{xy}}(\mathbf{x},\mathbf{y})$ is given as $f_{\mathbf{xy}}(\mathbf{x},\mathbf{y}) = f_{\mathbf{y}}(\mathbf{y}|\mathbf{x}) \cdot f_{\mathbf{x}}(\mathbf{x})$. From a finite set of N_v observations $\{\mathbf{x}_p,\mathbf{y}_p\}$ for $1 \le p \le N_v$ it is required to find a mapping $\overline{\mathbf{y}} = g(\mathbf{x},\boldsymbol{\omega})$ as close as possible to the desired mapping $\mathbf{y}$. With a physical constraint of no apriori knowledge of the distribution functions, presence of uncharacterized observation noise morphing the desired output, and requirement of generalizing the mapping function to unseen data, a generalization error measure (Er) can be written as:

$$Er = \int \theta \; f_{\mathbf{xy}}(\mathbf{x},\mathbf{y}) d\mathbf{x} d\mathbf{y} \tag{1}$$

where θ, a measure of loss estimate in the mapping, is a function of the difference between the desired output $\mathbf{y}$ and its estimate $\overline{\mathbf{y}}$ such as the squared error $\theta = (\overline{\mathbf{y}} - \mathbf{y})^2$.

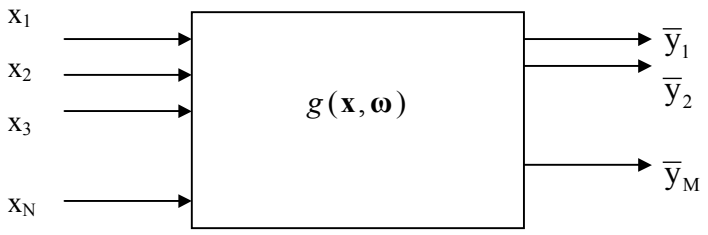

Fig. 2.1: MIMO function approximation representation.

The k[th] element of $\bar{\mathbf{y}}$, $\bar{y}_k$ can be written as some unknown function of input **x**

$$\bar{y}_k = g_k(\mathbf{x}, \boldsymbol{\omega}) \tag{2}$$

for $1 \leq k \leq M$. The corresponding Multiple Input Multiple Output (MIMO) mapping $\{ \mathbf{x} \in \Re^N \rightarrow \bar{\mathbf{y}} \in \Re^M \}$ is shown in figure 2.1. A classical parameterized form for the mapping is given by

$$\bar{y}_k = \sum_{i=0}^{L-1} \omega_{ki} \cdot f_i(\mathbf{x}) \tag{3}$$

for $1 \leq k \leq M$, where $f_i(\mathbf{x})$ are set of L basis functions and ω_{ki} is the corresponding unknown coefficient for i[th] basis function $f_i(\mathbf{x})$ and k[th] output $\bar{y}_k$. Thus the problem of function approximation is two-fold: (1) determination of an appropriate and minimal set of L basis functions and (2) determination of their corresponding coefficients. Parametric learning with known basis functions but unknown coefficients can be considered as a special case of the generalized non-parameterized representation. An equivalent representation of equation (3) is shown in figure 2.2 for output $\bar{y}_1$, and a similar representation can be found for all other outputs.

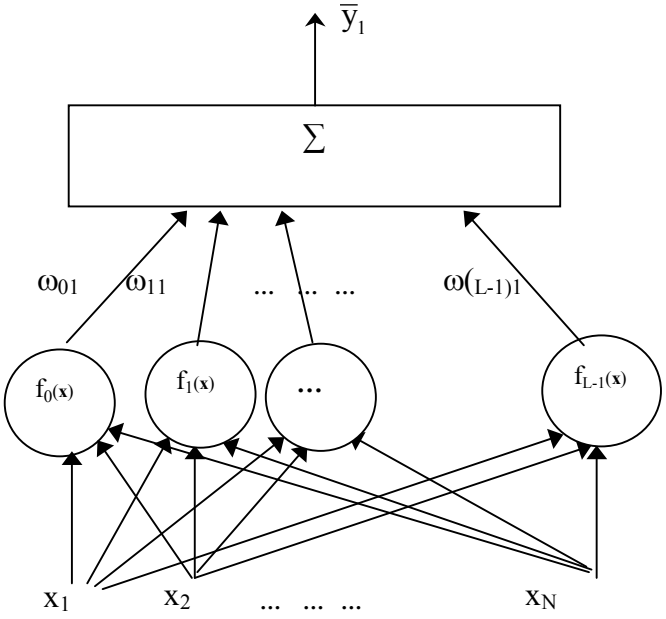

Fig. 2.2: Structural representation of function approximator.

For the k[th] output, the matrix representation of system in figure (2.2) for a dataset with N_v patterns can be written as

$$\begin{bmatrix} f_0(\mathbf{x}_1) & f_1(\mathbf{x}_1) & \bullet & f_{(L-1)}(\mathbf{x}_1) \\ f_0(\mathbf{x}_2) & f_1(\mathbf{x}_2) & \bullet & f_{(L-1)}(\mathbf{x}_2) \\ f_0(\mathbf{x}_3) & f_1(\mathbf{x}_3) & \bullet & f_{(L-1)}(\mathbf{x}_3) \\ \bullet & \bullet & \bullet & \bullet \\ f_0(\mathbf{x}_{N_v}) & f_1(\mathbf{x}_{N_v}) & \bullet & f_{(L-1)}(\mathbf{x}_{N_v}) \end{bmatrix} \begin{bmatrix} \omega_{k0} \\ \omega_{k1} \\ \omega_{k2} \\ \bullet \\ \omega_{k(L-1)} \end{bmatrix} = \begin{bmatrix} \bar{y}_{k1} \\ \bar{y}_{k2} \\ \bar{y}_{k3} \\ \bullet \\ \bar{y}_{kN_v} \end{bmatrix} \qquad (4)$$

where the i[th] basis function $f_i(\mathbf{x})$ takes values $f_i(\mathbf{x}_p)$ and output $\bar{y}_k$ takes values $\bar{y}_{kp}$ ($1 \le p \le N_v$, $0 \le i \le L\text{-}1$, $1 \le k \le M$). The set of basis functions can be pre-determined or developed during the training. Multi-layer Perceptrons (MLP)[4,5,20] uses non-linear squashing functions, like the sigmoidal function, as basis functions and have shown universal approximation capabilities. The Radial Basis Function (RBF) Network[4] use radial basis functions as basis functions for which the non-linear transformation occurs

at chosen center points for given dataset. Like the MLP, they too show similar approximation and interpolation properties. Genetic Algorithms, like the previous two networks develop the basis functions during training. Several pre-determined basis functions like the trigonometric, Legendre polynomials, algebraic polynomials, Chebyshev polynomials and Hermite polynomials have been shown to have universal approximation capabilities.

2.2 Mean Square Error (MSE) Criterion

A well-defined loss function for a given training dataset with N_v patterns is the optimal Minimum Mean Square Error (MMSE). For the system under consideration, equation (2-4), the MMSE estimate $\mathbf{y}_{MMSE}$ that minimizes the mean square error E_{MS}

$$E_{MS} = E[(\mathbf{y} - \mathbf{y}_{MMSE})^T (\mathbf{y} - \mathbf{y}_{MMSE})] \tag{5}$$

is given as

$$\mathbf{y}_{MMSE} = E[\mathbf{y}|\mathbf{x}] \tag{6}$$

where E is the expectation operator and the $\mathbf{y}_{MMSE}$ is the posteriori expected value of $\mathbf{y}$ for given $\mathbf{x}$. Equation (6) requires knowledge of apriori and likelihood densities which is not practically feasible in most approximation problems. However, it can be shown that an approximation network based on PBF's with certain assumptions can yield the MMSE criterion[21-23]. Consider the system with N inputs $\mathbf{x}$, M desired outputs $\mathbf{y}$ and $g(\mathbf{x},\boldsymbol{\omega})$ as network outputs where $\mathbf{w}$ is the weight coefficient matrix of the polynomial network which statistically span the range of the outputs to be estimated. Then the Mean Square Error (MSE) for the network under consideration is defined as:

$$E_t = \frac{1}{N_v}\sum_{p=1}^{N_v} || \, g(\mathbf{x}_p, \boldsymbol{\omega}) - \mathbf{y}_p \, ||^2 \tag{7}$$

where $(\mathbf{x}_p,\mathbf{y}_p)$ i=1,2,…,N_v is the training set representative of the true statistic from the population of **x** and **y**. The usage of MSE as the loss function is justifiable by the following lemma.

Lemma 1[21]: In the limit as N_v approaches ∞, the Mean Square Error (MSE) can be represented as an integral of the loss function over the joint density $f_{\mathbf{xy}}(\mathbf{x},\mathbf{y})$.

Proof: For N_v tending to ∞, by the Strong Law of Large Numbers[21,23], equation (7) tends to

$$E_{tL} = \lim_{N_v \to \infty} E_t = \iint || \, g(\mathbf{x}, \boldsymbol{\omega}) - \mathbf{y} \, ||^2 \; \mathrm{f}_{\mathbf{xy}}(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} \tag{8}$$

where as defined before $f_{\mathbf{xy}}(\mathbf{x},\mathbf{y})$ is the joint probability density function of **x** and **y**. *Q.E.D.*

From Bayes theorem, $f_{\mathbf{xy}}(\mathbf{x},\mathbf{y}) = f_{\mathbf{y}}(\mathbf{y}|\mathbf{x})\cdot f_{\mathbf{x}}(\mathbf{x})$ and equation (8) is written as:

$$E_{tL} = \int \mathrm{f}_{\mathbf{x}}(\mathbf{x}) \left[\int || \, g(\mathbf{x}, \boldsymbol{\omega}) - \mathbf{y} \, ||^2 \; \mathrm{f}_{\mathbf{y}}(\mathbf{y} \mid \mathbf{x}) d\mathbf{y} \right] d\mathbf{x} \tag{9}$$

To minimize (9) is equivalent to minimizing the quantity in the inner brackets, which is

$$E'_{tL} = \int || \, g(\mathbf{x}, \boldsymbol{\omega}) - \mathbf{y} \, ||^2 \; \mathrm{f}_{\mathbf{y}}(\mathbf{y} \mid \mathbf{x}) d\mathbf{y} \tag{10}$$

which will be minimized when $g(\mathbf{x},\boldsymbol{\omega}) = E[\mathbf{y}|\mathbf{x}]$ and $E[\mathbf{y}|\mathbf{x}] = \mathbf{y}_{MMSE}$ from (6). Thus a sufficiently complex $g(\mathbf{x}, \boldsymbol{\omega})$ polynomial network that successfully minimizes E'_{tL} approximates the Minimum Mean Square Estimator [23].

Networks like the MLP and RBF often employ an iterative approach to get close to a global minimum solution[24-27]. Polynomial Networks like the FLN that employ a flat-architecture are known to not suffer from problems of local minima[28] as their error landscape has only a global minimum.

2.3 Overspecialization

The property of the network to perform well on unseen data is called generalization. As the network is trained to minimize (7), it might run into over-specialization or memorization due to inadequate dataset size and distribution. In such cases even though the MSE for training data decreases there exists a good possibility that the network does not perform well on unseen data. Techniques like early stopping, cross-validation [29-30] have been employed to ensure a good level of generalization.

2.4 Polynomial Network Types

2.4.1. Functional Link Network

A FLN often has a fixed number of polynomial or trigonometric basis functions. For simplicity, a single-layer feed-forward 2^{nd} order FLN for 2 inputs x_1 and x_2, 1 output $\overline{y}_1$ and unknown weights w_k is shown in figure 2.3. The network can be extended similarly to show a $\Re^N \rightarrow \Re^M$ mapping.

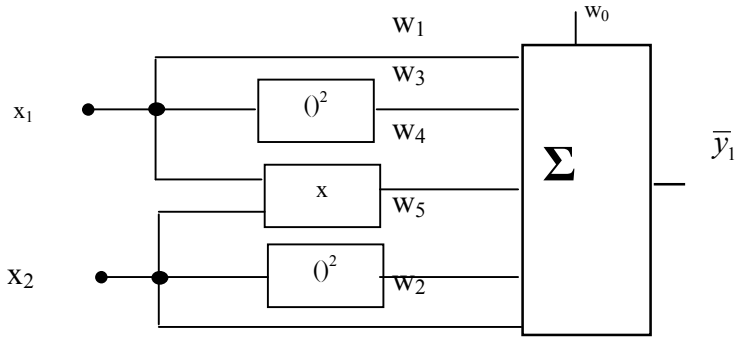

Fig. 2.3: A 2 input, 1 output degree 2 FLN.

As the non-linear learning for the FLN comes form the linear combinations of the pi elements, it is also called as a neural network with no hidden unit. FLN thus achieves function approximation in terms of the Kolmogorov-Gabor or Ivakhnenko polynomial

$$\overline{y}_k = \alpha_{k0} + \sum_{i=1}^{N} \alpha_{ki} \cdot x_i + \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{kij} \cdot x_i \cdot x_j + \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{l=1}^{N} \alpha_{kijl} \cdot x_i \cdot x_j \cdot x_l + \ldots \qquad (11)$$

for $1 \le k \le M$, where $\overline{y}_k$ is the k^{th} estimated output and α_k's are the corresponding weights. Let **X** be the column basis vector. Then the i^{th} polynomial basis function (PBF) element X_i is an element of the set $\{1, x_i, x_i \cdot x_j, x_i \cdot x_j \cdot x_k, \ldots\}$ for $1 \le i \le j \le k \ldots \le N$. Using polynomial basis functions as $f_i(\mathbf{x})$ in (3) and restricting (11) to L monomials, equation (11) is conveniently written as

$$\overline{y}_k = \sum_{i=0}^{L-1} w_{ki} \cdot X_i \qquad 1 \le k \le M \qquad (12)$$

where w_{ki} is the weight coefficient to i^{th} PBF X_i and k^{th} output $\overline{y}_k$ and the total number of basis functions L for D^{th} degree of approximation is given by

$$L = \binom{N+D}{D}. \qquad (13)$$

2.4.2. GMDH Network

A GMDH network generates combinations of degree 2 FLN network for each input pair. The useful basis functions inputs are then sorted and the network grows iteratively. This approach gives it a self-organized inductive learning property that helps in modeling higher order functions. A structural representation of multi-layered GMDH network is shown in figure 2.4.

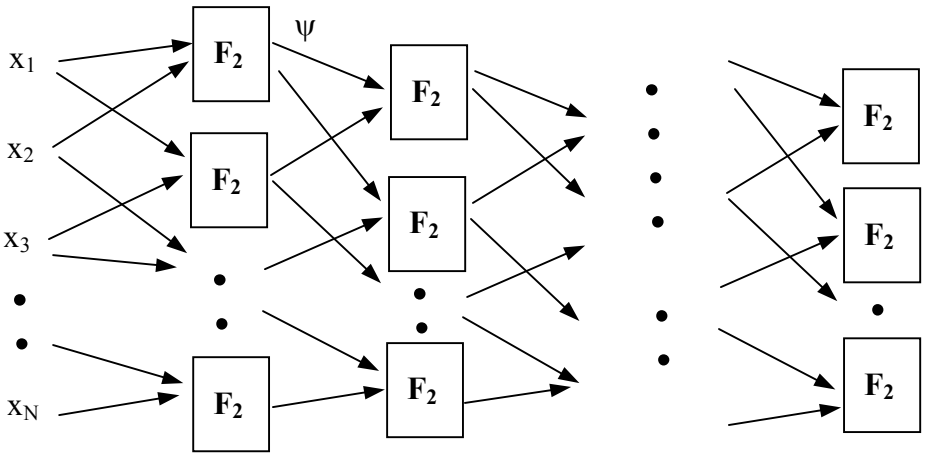

Fig. 2.4: Structural representation of GMDH network.

Output ψ of the basic processing block F_2 is given by the operation:

$$\psi = w_0 + w_1 \cdot x_i + w_2 \cdot x_j + w_3 \cdot x_i . x_j + w_4 \cdot x_i^2 + w_5 \cdot x_j^2 \quad (14)$$

for $1 \leq i,j \leq N$ .Starting with $N \cdot (N-1)/2$ combinations starting at first generation for a given training dataset, important basis functions are identified and sorted based on a regularity criterion (often root mean square) that best estimate the desired output and forms the inputs for second generation. This process is repeated for subsequent generations till the network begins to over-specialize or resultant estimation error for a

subsequent generation is higher then the previous one. Thus an n^{th} generation network approximates a function of 2^n order. The advantage of this network comes from its ability to model the higher order complexity. There are advanced versions of learning and growing networks using techniques like the genetic evolution that have shown some promising results for certain kind of problems[3]. For full-sized network to approximate the polynomial given in equation (11) the number of unknown weights to find for GMDH for 2^{nd} degree approximation is of order six times more than corresponding FLN for 2^{nd} degree. Figure 2.5 shows the corresponding plot for number or weights vs. number of inputs for 2^{nd} degree approximation comparison of GMDH and FLN. Also, implementation of GMDH for a MIMO mapping is a non-trivial task.

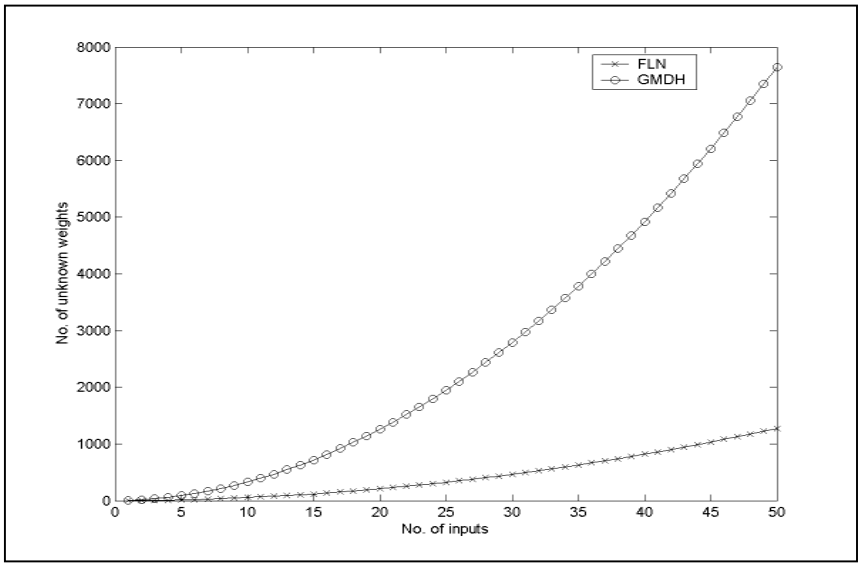

Fig. 2.5: Number of unknown weights comparison for D=2, GMDH and FLN

2.4.3. Pi-Sigma Network

A pi-sigma network is a two-layered network that achieves higher order learning as sums of higher order input correlations components[31] for a $\Re^N \rightarrow \Re^M$ mapping. The structural representation of the pi-sigma network is shown in figure 2.6.

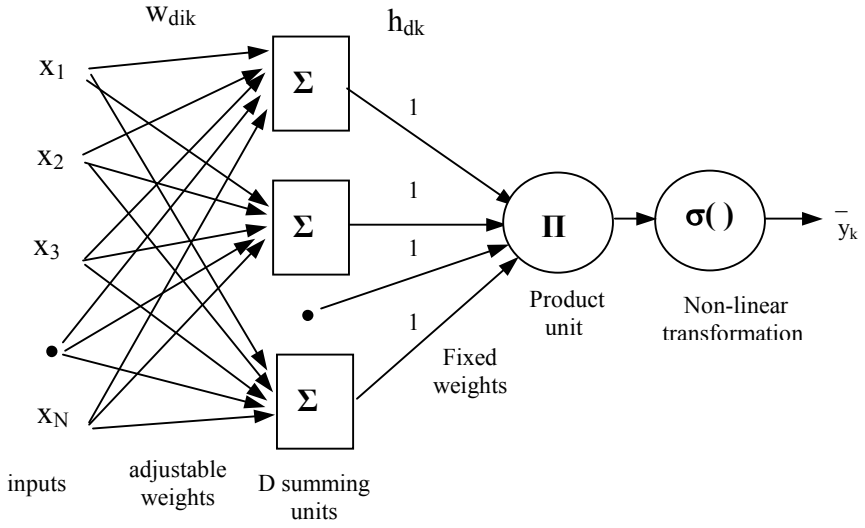

Fig. 2.6: Structural representation of a Pi-Sigma network

For D^{th} degree of approximation, the first-layer comprises of D summing units, the output of d^{th} unit of first-layer for k^{th} output h_{dk} is given by

$$h_{dk} = \sum_{i=1}^{N} w_{dki} \cdot x_i + w_{dk} \tag{14}$$

for $1 \le d \le D$, $1 \le i \le N$, $1 \le k \le M$, where w_{dki} is adjustable weight and w_{dk} is weight corresponding to threshold for d^{th} unit. The k^{th} estimated output is given by

$$\bar{y}_k = \sigma\left(\prod_{d=1}^{D} h_{dk} \right) \tag{15}$$

where σ() is any suitable non-linear squashing function. Clearly an important advantage of this network is for higher degree approximation the number of unknown weights is only M·D· (N+1). Table 2.1 shows the comparison for number of unknowns for a Sigma-pi and FLN network. However, pi-sigma networks are not universal approximators and they trade-off approximation capabilities to network size. Ridged-Polynomial Networks[32] which are incremental networks obtained from varying degrees of pi-sigma networks have shown better approximation accuracy.

Table 2.1 Number of Weights Comparison for FLN and Pi-sigma Networks

Degree of approximation	Number of Weights (free parameters)			
	FLN		Pi- sigma	
	N=10	N=15	N=10	N=15
1	11	16	11	16
2	55	120	22	32
3	286	816	33	48
4	1001	3876	44	64

2.5 Properties of Polynomial Networks

2.5.1. Advantages of Polynomial Networks

Few advantages of the Polynomial Networks over other networks are:

1. Polynomial Networks (PN) use PBF's which unlike sigmoidal functions are unbounded and hence provide faster learning.
2. PN compared to other networks are easily tractable and extremely informative as they provide an understanding of representation of PBF's and hence exhibit open-box model behavior.

3. PN's are fast and computationally efficient as compared to other networks like MLP as they can be realized using multiplication and addition operators and have smaller storage capacity.
4. Due to the flat-architecture PN's like the FLN do not suffer from local-minima problems.

2.5.2. Disadvantages of Polynomial Networks

Polynomial Networks suffer from a few disadvantages when compared to other networks like the MLP:

1. The computational complexity of PN's increases exponentially with increases in degree of approximation. This problem is often referred as the curse of dimensionality. For a D^{th} degree approximation of N inputs, the number of required basis function (L) as in equation (13) which leads to combinatorial explosion with increase in either number of inputs and degree of approximation D. For example, for N = 25 and D = 2 the number of basis functions L required is 351 but for D = 4, L = 23751 and for D = 8, L = 13,884,156. For a MLP, the network grows in size and complexity by adding more hidden units and since non-linear transforms occur at the sigmoid a relatively few hidden units may be possible to achieve desired degree of complexity.
2. Highly non-linear functions may require very high degree polynomials and may not be well approximated, for these MLP might perform better.

GMDH network and GMDH-like networks incrementally grow the network for higher order learning to overcome the combinatorial explosion, but the approach is heuristic based and often results in presence of unwanted functions or extremely large sized networks. Work based on GMDH and FLN employing Genetic Algorithms have shown some good results[3,33], but they lose the properties of polynomial networks.

CHAPTER 3

THE ORDERED FUNCTIONAL LINK NETWORK

It is desired to retain the MIMO universal approximation capabilities of the FLN and avoid any disadvantages. Applying an Iterative Gram-Schmidt procedure to a FLN a multi-layered, regularized incremental growth, and well-structured network is achieved. The FLN has ordered basis functions and hence is called the Ordered Functional Link Network (OFLN). The OFLN reduces the possibility of combinatorial explosion and achieves better approximation capabilities. In this chapter the standard Gram-Schmidt procedure and its numerically stable version is reviewed and an implementation algorithm for the OFLN is presented.

3.1 The Gram-Schmidt Orthonormalization Procedure

The Gram-Schmidt procedure maps a set of linearly dependent vectors to an orthonormal basis vector set in Euclidian or any inner product space. It is a well-known standard numerical method [34] and has been used to find optimum choice of Radial centers for RBF [35], fast computation of weights of RBF network [36], pruning of MLP [37], feature selection in piecewise classifier [38]. What few authors did not realize is that using the Gram-Schmidt procedure allows ordering the basis function in order of their contribution to minimize the MSE. Using this the network can be represented as a monotonically non-increasing function of addition of basis functions

and achieve a faster rate of convergence. Also, orthonormalizing linearly dependent vectors results in a zero vector and is an important step in pruning useless basis functions. These desirable properties along with the effective representation, system re-transformation and a fast and distributed iterative solution make it a better candidate over other learning algorithms.

Consider a vector $\mathbf{X}$ $(X_0,X_1,X_2,\ldots\ \ldots,X_{L-1})$ whose elements are basis functions and its corresponding orthonormal mapping is $\mathbf{X^o}$ $(X^o_0,X^o_1,X^o_2,\ldots\ \ldots,X^o_{L-1})$ where both $\mathbf{X}$ and $\mathbf{X^o} \in \Re^L$. By the definition of orthonormality:

$$<X^o_i\, X^o_j> = 0 \qquad \text{for} \quad i \neq j \tag{16}$$

$$= 1 \qquad \text{for} \quad i=j \quad i,j \in (0,L-1) \tag{17}$$

where, $< X^o_i\, X^o_j >$ is defined as the correlation for N_v patterns.

$$< X^o{}_i X^o{}_j >= \frac{1}{N_v}\sum_{p=1}^{N_v} X^o{}_{ip} \cdot X^o{}_{jp} \tag{18}$$

Here X^o_{ip} corresponds to the p^{th} value of i^{th} orthonormal function X^o_i. Similarly, $<X_iX^o_j>$ and the mode of $\| X_i \|$ is defined as

$$< X_i X^o{}_j >= \frac{1}{N_v}\sum_{p=1}^{N_v} X_{ip} \cdot X^o{}_{jp} \quad \text{and} \quad \| X_i \| = \sqrt{\frac{1}{N_v}\sum_{p=1}^{N_v} X^2{}_{ip}} \tag{19}$$

Then by the standard Gram-Schmidt procedure, X^o_k are calculated as

$$X^o{}_0 = \frac{X_0}{\| X_0 \|} \tag{20}$$

$$X^o{}_1 = \frac{X_1 - < X_1 X^o{}_0 > X^o{}_0}{\| X_1 - < X_1 X^o{}_0 > X^o{}_0 \|} \tag{21}$$

…

$$X^o{}_k = \frac{X_k - \sum_{i=0}^{k-1} < X_k X^o{}_i > X^o{}_i}{\| X_k - \sum_{i=0}^{k-1} < X_k X^o{}_i > X^o{}_i \|} \tag{22}$$

for $1 \leq k \leq L\text{-}1$.

3.1.1. An Iterative Gram-Schmidt Procedure

Inherently the Gram-Schmidt procedure is not numerically stable due to introduction of rounding errors when implemented on a computer. An iterative solution to is given here. **A** represents a lower triangular L x L orthonormal transformation matrix such that:

$$\mathbf{X^o = A \bullet X} \tag{23}$$

$$A = \begin{bmatrix} a_{00} & 0 & 0 & 0 \\ a_{10} & a_{11} & 0 & 0 \\ . & . & . & . \\ a_{(L-1)0} & a_{(L-1)1} & . & a_{(L-1)(L-1)} \end{bmatrix} \tag{24}$$

Thus the m^{th} orthonormal function can be obtained from **X** and **A** by

$$X^o{}_m = \sum_{i=0}^{m} a_{mi} \cdot X_i \tag{25}$$

for $0 \leq m \leq L\text{-}1$. Let the auto-correlation function r_{ij} be defined as:

$$r_{ij} = \frac{1}{N_v} \sum_{p=1}^{N_v} X_{ip} \cdot X_{jp} \tag{26}$$

for $0 \leq i,j \leq L\text{-}1$. Then from (23-26),

$$a_{00} = \frac{1}{|| X_0 ||} = \frac{1}{(r_{00})^{\frac{1}{2}}} \tag{27}$$

$$X^o_0 = a_{00} \cdot X_0 \tag{27}$$

$$X^o_1 = \sum_{i=0}^{1} a_{1i} \cdot X_i = a_{10} \cdot X_1 + a_{11} \cdot X_1 \tag{28}$$

Let X^0_1 be equal to

$$X^o_1 = \frac{\phi_1}{|| \phi_1 ||} \tag{29}$$

Then $\phi_1 = X_1 - \beta_0 \cdot X^o_o = X_1 - \beta_0 \cdot a_{00} \cdot X_o$ (30)

where $\beta_0 = \langle X^o_o \cdot X_1 \rangle = a_{00} \cdot r_{01}$ (31)

Writing ϕ_1 as

$$\phi_1 = \sum_{k=0}^{1} \gamma_k X_k = \gamma_0 X_0 + \gamma_1 X_1 \tag{32}$$

Here, $\gamma_0 = -\beta_0 \cdot a_{00}$ (31)

$$\gamma_1 = 1 \tag{32}$$

Also, $|| \phi_1 || = || < X_1 - \beta_0 \cdot X^o_0, X_1 - \beta_0 \cdot X^o_0 > ||$ (33)

$$\therefore || \phi_1 || = \left[r_{11} - \beta_0^2 \right]^{\frac{1}{2}} \tag{34}$$

Equating (28) and (32)

$$X^o_1 = a_{10} \cdot X_1 + a_{11} \cdot X_1 = \frac{\gamma_0 X_0 + \gamma_1 X_1}{\left[r_{11} - \beta_0^2 \right]^{\frac{1}{2}}} \tag{35}$$

$$a_{10} = \frac{-\beta_0 a_{00}}{\left[r_{11} - \beta_0{}^2 \right]^{\frac{1}{2}}} \tag{36}$$

$$a_{11} = \frac{1}{\left[r_{11} - \beta_0{}^2 \right]^{\frac{1}{2}}} \tag{37}$$

An iterative approach for finding the a_{ij} coefficients can be extended as follows:

For $1 \le m \le L\text{-}1$, perform the following operations

$$\beta_i = \sum_{q=0}^{i} a_{iq} \cdot r_{JqJm} \qquad \text{for} \qquad 0 \le i \le m\text{-}1 \tag{38}$$

$$\gamma_m = 1 \tag{39}$$

$$\gamma_k = -\sum_{i=k}^{m-1} \beta_i \cdot a_{ik} \qquad \text{for} \qquad 0 \le k \le m\text{-}1 \tag{40}$$

$$a_{mk} = \frac{\gamma_k}{\left[r_{J_m J_m} - \sum_{i=0}^{m-1} \beta_i{}^2 \right]^{\frac{1}{2}}} \qquad \text{for} \qquad 0 \le k \le m \tag{41}$$

3.1.2. Solving for the Orthonormal System Weights

When the basis functions **X** are transformed to $\mathbf{X^o}$, the system is mapped into new weights w^o_{ki} for the k^{th} estimated output $\overline{y}_{kp}$ and i^{th} orthonormal basis function X^o_i for given training dataset with N_v patterns.

$$\overline{y}_{kp} = \sum_{i=0}^{L-1} w^o{}_{ki} \cdot X^o{}_{ip} \tag{42}$$

for $1 \le p \le Nv$. The MSE in terms of the new weights for N_v desired values of y_k can be written as

$$E_k = \frac{1}{N_v}\sum_{p=1}^{N_v}\left[y_{kp} - \sum_{i=0}^{L-1} w^o{}_{ki} \cdot X^o{}_{ip} \right]^2 = \left\langle y_k - \sum_{i=0}^{L-1} w^o{}_{ki} \cdot X^o{}_i , y_k - \sum_{i=0}^{L-1} w^o{}_{ki} \cdot X^o{}_i \right\rangle \tag{43}$$

for $1 \le k \le M$. Using the definition of orthonormal functions (16-17) and expanding (43)

$$E_k = \langle y_k , y_k \rangle - \sum_{i=0}^{L-1} (w^o{}_{ki})^2 \tag{44}$$

Taking partial derivative w.r.t. the variable $w^o{}_{km,}$ L equations in L unknowns are obtained, thus there would be a unique solution with only a global minimum. Using (25) the orthonormal weights are given by

$$w^o{}_{km} = \sum_{i=0}^{m} a_{mi} \cdot c_{ki} \tag{45}$$

for $1 \le k \le M$, $0 \le m \le L\text{-}1$, where c_{ki} is the cross-correlation function defined as

$$c_{ki} = \langle y_k , X_i \rangle = \frac{1}{N_v}\sum_{p=1}^{N_v} y_{kp} \cdot X_{ip} \tag{46}$$

for $1 \le k \le M$, $0 \le i \le L\text{-}1$, where y_{kp} corresponds to the p^{th} pattern of output y_k.

3.1.3. Re-mapping System Weights from Orthonormal System

To understand the mapping relationship in terms of the original system and to avoid additional computation of orthonormalizing the basis functions for validation and real-time processing, it is important to represent the system in terms of the original weights. An efficient mapping orthonormal weights to the original system weights is achieved by equating (12) and (43) and then substituting for $\mathbf{X^o}$ from (25)

$$w_{ki} = \sum_{j=0}^{L-1} w^{o}{}_{kj} \cdot a_{ij} \tag{47}$$

Lemma 2: If Gram-Schmidt procedure is applied to a linearly dependent vector sequence, X_m, then the m^{th} vector is a zero vector.

Proof: Consider X_m that is linearly dependent on other vectors up to m-1, then by definition of linearity there exists at least one non-zero λ_i such that

$$X_m = \sum_{i=1}^{m-1} \lambda_i \cdot X_i \tag{48}$$

Then X^o_m from (22) is given as

$$X^o{}_m = \frac{X_m - \sum_{i=0}^{m-1} < X_m X^o{}_i > X^o{}_i}{\| X_m - \sum_{i=0}^{m-1} < X_m X^o{}_i > X^o{}_i \|} \tag{49}$$

Substituting (48) in (49), the numerator is then

$$X_m - \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_j < X_j X^o{}_i > X^o{}_i \tag{50}$$

$$= X_m - \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_j < X_j X^o{}_i > \sum_{k=0}^{i} a_{ik} \cdot X_k \tag{51}$$

$$= X_m - \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_j \sum_{k=0}^{i} a_{ik} < X_j X^o{}_i > X_k$$

Apply change of variable,

$$= X_m - \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_j < \sum_{k=0}^{j} a_{jk} \cdot X_j X^o{}_i > X_j \tag{52}$$

$$= X_m - \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_j < X^o{}_j X^o{}_i > X_j \qquad (53)$$

where (25) is substituted in (51) and again used in (52). Using the definition of orthonormal vectors as in (16-17), the inner product will be 1 only for i = j and otherwise. Then equation (53) reduces to

$$= X_m - \sum_{j=0}^{m-1} \lambda_j X_j \qquad (54)$$

which by definition of X_m as in (48) equals zero. *Q.E.D.*

3.2 Implementation of the Ordered Functional Link Network (OFLN)

Consider the FLN discussed in section 2.3.1 with N inputs and M outputs and N_v training patterns, $\{\mathbf{x}_p, \mathbf{y}_p\}_{p=1,N_v}$. Assume that the N-dimensional input vector **x** has mean vector **m** and a vector **σ** of standard deviations. In order to increase the likelihood that the PBF's in **X** are well behaved, normalize the elements of **x** as

$$x_{np} \leftarrow (x_{np} - m_n) / \sigma_n \qquad (55)$$

for $1 \le n \le N$, $1 \le p \le N_v$. The bias term, x_o is defined to be 1. A loss measure of approximation for the OFLN, the MSE, from (12) is written as

$$E_t = \sum_{k=1}^{M} E_k = \sum_{k=1}^{M} \frac{1}{N_v} \sum_{p=1}^{N_v} \left[y_{kp} - \sum_{i=0}^{L-1} w_{ki} \cdot X_{ip} \right]^2 . \qquad (56)$$

where, as in equation (26) y_{kp} is the p^{th} desired value for k^{th} output y_k, X_{ip} is the p^{th} value of i^{th} PBF X_i, and w_{ki} denotes the unknown weight from i^{th} PBF X_i to k^{th} output y_k.

3.2.1. Notation and Representation

Limiting the degree of approximation D for the FLN can contain the problem of combinatorial explosion but this also limits its ability to model complex functions. A better approach is to grow the network with most useful basis functions forming the higher degree terms. If the elements of **X** are to be in descending order of their usefulness a method is needed for generating these efficiently, in any possible order. Consider an L by (D+1) position matrix **K**, where D is the desired degree of approximation, whose i^{th} row specifies how to generate the PBF X_i. For element K(i,j), the ranges of i and j are $0 \le i \le L-1$ and $0 \le j \le D$. The i^{th} PBF X_i, with K(i,D) denoting its degree, is defined as

$$X_i = \prod_{j=1}^{K(i,D)} x_{K(i,j-1)} \tag{57}$$

The first basis function denotes the bias term and is fixed as $X_o = 1$. Thus **X** is generated from **K** and the normalized input vector **x**. Using the iterative Gram-Schmidt procedure **X** is orthonormalized to $\mathbf{X^o}$. A structural representation of FLN with orthonormal transformation for single output is shown in figure 3.1.

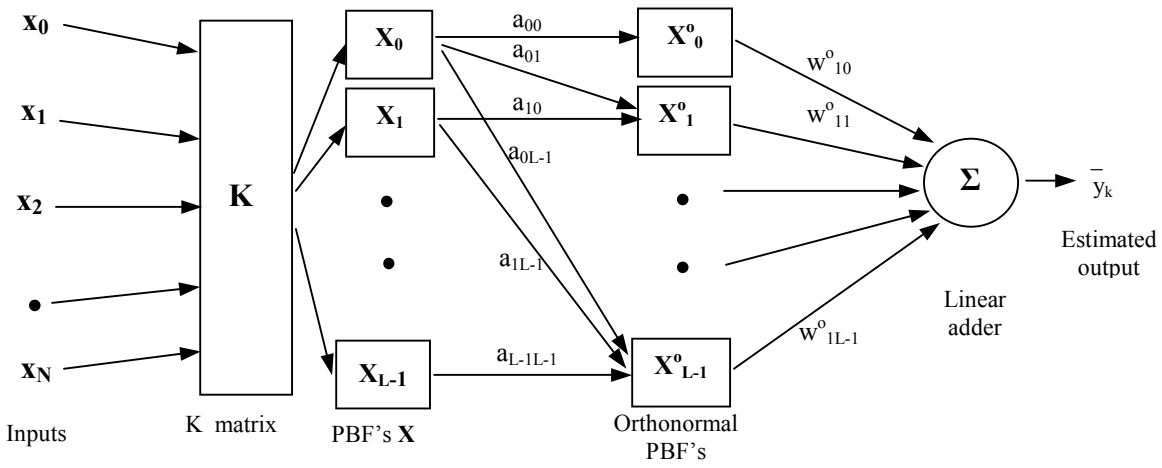

Fig 3.1: Structural representation of OFLN for single output.

The output $\overline{y}_k$ as shown in fig. 3.1 for k=1 and its transformed weight w^o_{ki} corresponding to i^{th} orthonormal vector X^o_i is given by

$$\overline{y}_{kp} = \sum_{i=0}^{L-1} w^o{}_{ki} \cdot X^o{}_{ip} \tag{58}$$

for $1 \leq k \leq M$. Let the elements of array **J** index basis functions according to their usefulness, i.e. if $J_0 = 3$ and $J_3 = 8$, then the 1^{st} and 4^{th} most useful basis functions that contribute the most to reduce the MSE are respectively X_3 and X_8. Thus **J** points to ordered PBF's that effectively contribute to reduce the MSE.

3.2.2. Training the OFLN

The general approach for training the OFLN is to iteratively generate **K** and **J** for higher and higher degrees, D, finding the basis function coefficients each time using

Gram-Schmidt procedure. In each iteration, redundant and useless basis functions are eliminated, in order to prevent combinatorial explosion in the subsequent iterations.

3.2.2.1 Degree D up to One

In the first OFLN training iteration, the inputs are ordered according to their usefulness. In this case, L = N+1 and **K** and **J** are initialized as

$K(i,0) = i \quad K(i,1) = 1$

$J_i = i$ (59)

for $0 \leq i \leq N$. Following the basis function definition in (57), (59) indicates that the maximum degree is D=1, the first basis function referred by J_0 and K(0,0) is the constant 1 corresponding to D=0, and the remaining basis functions are inputs. Starting with initial values in (59) it is desired that **J** point to linearly independent inputs in order of their contribution to reduce the MSE. The m^{th} orthonormal basis function X^o_m using index vector **J** from (59) and (25) is now re-written as

$$X^o_m = \sum_{i=0}^{m} a_{mi} \cdot X_{J_i} \quad . \tag{60}$$

Iterative Gram-Schmidt procedure is applied to find the unknown coefficients for the OFLN, a modification equations (38-45) using **J**:

for m=0

$$X^o_0 = a_{00} \cdot X_{J_0} \tag{61}$$

$$a_{00} = \frac{1}{|| X_{J_0} ||} = \frac{1}{r_{J_0 J_0}} \tag{62}$$

for $1 \le m \le L-1$, repeat the following operations

$$\beta_i = \sum_{q=0}^{i} a_{iq} \cdot r_{JqJm} \qquad 0 \le i \le m-1 \qquad (63)$$

$$\gamma_m = 1 \qquad (64)$$

$$\gamma_k = -\sum_{i=k}^{m-1} \beta_i \cdot a_{ik} \qquad 0 \le k \le m-1 \qquad (65)$$

$$a_{mk} = \frac{\gamma_k}{\left[r_{J_m J_m} - \sum_{i=0}^{m-1} \beta_i^{\,2} \right]^{\frac{1}{2}}} \qquad 0 \le k \le m \qquad (66)$$

$$w^{o}_{km} = \sum_{i=0}^{m} a_{mi} \cdot c_{kJ_i} \qquad 1 \le k \le M \qquad (67)$$

Lemma 3: If any input x_m is linearly dependent on other inputs then higher order basis functions that include x_m can be expressed using basis functions of the same degree that do not include x_m.

Proof: For any input x_m that is linearly dependent on other inputs there exists at least one non-zero λ_i such that

$$x_m = \sum_{i=1, i \ne m}^{N} \lambda_i \cdot x_i \qquad (68)$$

Consider a degree D basis function with dependent input x_m raised to the d^{th} power. We have

$$x_m^{\,d} \prod_{n=1, k(n) \ne m}^{D-d} x_{k(n)} = \left(\sum_{i=1, i \ne m}^{N} \lambda_i \cdot x_i \right)^d \prod_{n=1, k(n) \ne m}^{D-d} x_{k(n)} \,. \qquad (69)$$

In (69), the right hand side has no x_m and degree D is also unchanged. *Q.E.D.*

Using lemmas 2 and 3, for $1/a_{mn} \rightarrow 0$, the m^{th} linearly dependent function can be eliminated as below

$$J_i = J_{i+1} \quad \text{for } m \leq i < L-1 \tag{70}$$

$$L \leftarrow L-1 \tag{71}$$

Denote the 2^{nd} term in the MSE for the orthonormal system in (44) as P_i associated with i^{th} orthonormal basis function X^o_i

$$P_i = \sum_{k=1}^{M} \left[w^o{}_{ki} \right]^2 . \tag{72}$$

Equation (72) gives us an important order relationship that defines the usefulness of a basis function to reduce the MSE. The MSE will be minimized when () is maximized. A physical interpretation of (44) simply means that the basis function with maximum value of absolute weight contributes the most to reduce the MSE and this interpretation is possible because the basis functions are orthonormalized. Thus, the desired new order of basis functions **J** that reduce the MSE is obtained by maximum value of P_i and is given by

$$P_{J_0} \geq P_{J_1} \geq .....P_{J_{L-2}} \geq P_{J_{L-1}} . \tag{73}$$

Using (47), the orthonormal weights can be transformed back to original weights. For D=1, **J** gives the ordered basis functions for a linear network. If a first order approximation is required, then a reordered **K** based on **J** and weights from (73) could be saved and it represents the OFLN of degree 1. Zero order function approximation is a special case with L = 1 and can be solved for as above.

3.2.2.2 Degree D Greater Than One and Stopping Criterion

The network grows iteratively for each degree up to desired D. **K** from the previous case is reordered based on **J** such that it corresponds to only essential PBF's. Indices of candidate basis functions of degree under consideration are generated from combinations of **K** and appended to it. One approach to ensure only unique higher degree functions are added is to sort the component inputs, as specified in a given new row of K. The row is kept only if it does not duplicate one of the rows above. Higher order basis functions **X** are then generated from (6). Equations (14)-(19), (22)-(24), (26)-(27) are repeated with the value of L being the row count of **K** for each degree. Higher order linearly dependent functions can be eliminated by extending the Lemma 3 for higher degree PBF's. As a control or stopping criterion, number of PBF's degree under consideration can be limited by stopping at a given maximum number of PBF's L_{max}. Alternately, another criterion can be when the relative percentage change in error for adding a PBF is less than a user-chosen value $\Delta\varepsilon$.

3.2.3. Fast Validation and Network Sizing

For the OFLN a one-pass measurement of validation error for network size up to L is possible. The validation dataset is normalized with known values of **m** and **σ**. **X, X^o** are generated using (6) and (13) respectively. For a network of size k with N_{vt} validation patterns, the total validation MSE (E_{vk}) is given by

$$E_{vk} = \frac{1}{N_{vt}} \sum_{p=1}^{N_{vt}} \left[\sum_{i=1}^{M} \left(y_{ip} - \sum_{j=0}^{k-1} w^{o}{}_{ij} \cdot X^{o}{}_{jp} \right)^2 \right] \tag{74}$$

for $1 \leq k \leq L$. For the pattern number p, the quantity in the inner brackets of (74) can be evaluated for all values of k. Hence E_{vk} can be updated for all additive sets of basis functions ($1 \leq k \leq L$) in a single pass through the data thereby providing fast validation and network sizing.

CHAPTER 4

OFLN CLASSIFIER

A Classifier maps an input feature vector **x** to discrete label or class **i**. Alike the function approximation, classification problem can be supervised and unsupervised, thus a classifier will learn generalized mapping rules from a given training dataset. Classification has immense applications in receiver design (communication engineering), econometrics, computer vision, bioinformatics, speaker dependent and speaker independent speech recognition, industrial engineering, etc [39-41].

An input feature in a classifier is mapped to finite discriminant functions that form an enclosed hypersurface of disjoint sets, each set representative of the class label. The classification error, the number of patterns not classified correctly, is minimum for an optimal Bayes classifier that tends to determine the aposteriori class probability for a given feature vector [39]. The Bayes discriminant functions require apriori knowledge of the class probabilities and likelihood density functions that in most practical classification problems are not available or difficult to analyze. Hence other types of classifiers that perform as well as the Bayes classifier have been researched and studied. The Bayes-Gaussian Classifier, Nearest Neighbor Classifier, Feed-forward Network, Piecewise Linear Network and MLP as Classifier [39,41-43] perform well under certain scenarios and are often employed in practice.

4.1 Regression Based Classifiers

A regression-based network, which successfully minimizes MSE, under certain conditions has been shown to approximate the optimal Bayesian discriminant function[44]. A proof of the MSE approaching the Bayesian discriminant for a feature set with multiple classes is referred to in Appendix A. Thus a regression network with some modifications performs as a very useful classifier, few examples of such classifiers are the Feed-forward networks like MLP and PLN.

4.1.1. Classifier Notations and Overview

For a classification problem under consideration let $\mathbf{x}$ ($\mathbf{x} \in \Re^N$) be the input feature vector, N_c be total number of classes, N_v the total number of training patterns. Let the desired output discriminant functions be represented as $y_{ip,}$ the i^{th} desired output for p^{th} pattern. It is required that y_{ip} approximate the optimal Bayesian discriminant function $d_i(\mathbf{x})$ given by

$$d_i(\mathbf{x}) = P(i|\mathbf{x}) \tag{75}$$

where $P(i|\mathbf{x})$, i.e. the posteriori probability that given a feature vector $\mathbf{x}$ it belongs to class i. To maximize the separation distance between 2 classes, y_{ip} for a given input feature $\mathbf{x}$ belonging to class j is initially defined as

$$y_{ip} = 2 \cdot c \cdot \delta(i-j) - c \tag{76}$$

where c is a positive real number. Let $\overline{y}_{ip}$ be the estimated output discriminant function corresponding to desired output $y_{ip,}$ then the MSE for the regression network given training patterns N_v is then given by

$$E_t = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^{N_c} \left(y_{ip} - \overline{y}_{ip} \right)^2 \tag{77}$$

The training classification error (P_{et}) is ratio of patterns of the given dataset that are not classified properly. Given the proof in Appendix A and for training patterns $N_v \rightarrow \infty$, it is sufficient enough to minimize (77) and achieve reasonably small classification error.

The output discriminant functions on adequate training represents the aposterior probabilities P(i|**x**). However, the output functions can have values less than 0 and greater than 1 and thus the interpretations of the above statement are limited as the classification error is unaffected by the stochastic constraints $0 \leq P(i|\mathbf{x}) \leq 1$ and $\sum_{i=1}^{N_c} P(i \mid x) = 1$ which might not hold true. The standard error criterion can be relaxed so as to primarily improve the classification error if theoretical links to Bayes decision rule are not broken. This can be achieved by using an Output Reset (OR) algorithm [45] as described briefly here.

4.2 The Output Reset Algorithm

E_t in equation (77) can be thought of as a residual error which is accumulated for each of N_c discriminant function for N_v patterns. A class decision i is made for $\max_i \left(y_{ip} - \overline{y}_{ip} \right)^2$. The MSE is redundantly higher because of 1) patterns belonging to the correct and incorrect class with discriminant functions of similar sign of desired output but of greater magnitude 2) patterns for incorrect class with discriminant functions having magnitude less than the desired output. Thus the residual error

contains at least two types of biases removal of which would not immediately affect the classification error. The biases can be removed by introducing a new desired output y'_{ip} such that

$$y'_{ip} = y_{ip} + a_p + d_{ip} \tag{78}$$

where a_p is the additive bias common to each pattern and d_{ip} is the bias for each discriminant function and pattern. The corresponding MSE E'_t is given by

$$E'_t = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^{N_c} \left(y'_{ip} - \overline{y}_{ip} \right)^2 \tag{79}$$

It is required to find optimum values of a_p and d_{ip} under the following conditions:

1. The difference $d_{i_c p} - d_{i_d p}$ must be greater than or equal to zero, where i_c denotes the correct class and i_d denotes an incorrect class, otherwise d_{ip} could cancel y_{ip} thus persuading learning algorithms to drive weights toward zero in an effort to minimize E'_t.
2. Each change made to a_p, d_{ip} and y'_{ip} through changes in network weights must reduce E' or at least keep it unchanged.

Equating the gradient of E'_t to 0 and solving for a_p yields

$$a_p = \frac{1}{N_c} \sum_{i=1}^{N_c} \left[y_{ip} - \overline{y}_{ip} - d_{ip} \right] \tag{80}$$

When a_p is added to each desired output $y_{ip,}$ the distances between correct class residual and incorrect class residual remains unchanged and hence classification error is unchanged. Also, a_p would minimize E'_t, both condition 1 and 2 are satisfied.

Similarly d_{ip} can be found by minimizing square error term $[d_{ip} + a_p + y_{ip} - \bar{y}_{ip}]^2$ to give $d_{ip} = [y_{ip} - \bar{y}_{ip} - a_p]$. To satisfy condition 1 d_{ip} is constrained such that for a correct class, $d_{i_c p} \geq 0$, and for an incorrect class, $d_{i_d p} \leq 0$. An iterative solution to the problem of finding a_p and t_{ip} along with a closed form representation is discussed in [45].

4.3 OFLN Classifier Implementation

The OFLN minimizes the MSE for a given training dataset by mapping the N_c classes to fixed points in $\Re^{Nc}$ Euclidian space and forms a regression-type classifier. A representation of the OFLN classifier is shown in figure (4.1), here the number of estimated outputs for the network equals the number of classes N_c.

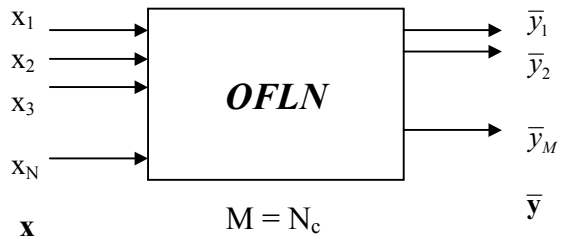

Fig. 4.1: Representation of OFLN Classifier.

4.3.1. Algorithm for Implementation of the OFLN classifier

1. The OFLN is trained to minimize the MSE by setting the desired output $\mathbf{y}_{kp}$ as defined in (76) for the given N_v training patterns. The iterative Gram-Schmidt

learning algorithm, as in section 3.2, is used to obtain the orthonormal PBF's $\mathbf{X}^o$, the **A** matrix and the weights w^o_{ki}.

2. For a training pattern under consideration, calculate the discriminant functions $\bar{y}_{kp}$ ($1 \le k \le M$) from $\mathbf{X}^o$ and w^o_{ki} using equation (58).
3. Use OR to find the new desired outputs y_k for the training pattern using information of the correct class label i_c and discriminant function $\bar{y}_k$.
4. Update the new cross correlation function for the desired outputs y_k and $\mathbf{X^o}$ using equation (46)
5. Repeat steps 2-4 for each training pattern.
6. Find the new orthonormal system weights w^o_{ki} using equation (45), elements of **A** are fixed for OR algorithm and obtained from the step 1.
7. Find new discriminant functions $\bar{y}_k$ for each pattern using $\mathbf{X^o}$ PBF's and corresponding weights w^o_{ki} and determine the estimated correct class i'_c for the incremental basis functions:

$$i'_c = \underset{k}{\operatorname{argmax}}\, \bar{y}_{kp}(m) = \underset{k}{\operatorname{argmax}} \sum_{j=0}^{m} w^o{}_{kj} \cdot X^o{}_{jp} \qquad (81)$$

for $1 \le k \le M$

8. Increment the classification error count $P_e(m)$ if $i'_c \neq i_c$ for $0 \le m \le L-1$
9. Repeat steps 7-8 for each training pattern.
10. Repeat steps 2-9 for the desired N_{it} iterations. Save the values of $P_e(m)$ and corresponding weight matrix which is used to measure the validation error for unseen data.

As in the case of function approximation, early stopping and validation criterion are used to determine a stopping criterion. Training and validation classification error performance for incremental degree of OFLN, incremental network size and incremental OR iterations can be studied to determine the most appropriate classification network for a particular problem.

CHAPTER 5

SIMULATION EXAMPLES

Simulation examples for OFLN function approximation and classification are presented where performance is compared with that of other networks. Algorithms for training, validation and processing datasets were implemented in MS C++ 6.0 for Win32 upward systems.

5.1 Function Approximation

Graphs for learning non-polynomial sine function and single-mode modified rastrigin function show the function approximation capabilities of the OFLN. As in [31] 10 data values in the interval [0:1] are used for learning sine function (fig. 5.1). The approximated rastrigin function (given by $20+x_1^2+ x_2^2 -10(\cos(0.1\pi x_1) + \cos(0.1\pi x_2))$ in fig. 5.2 shows an important advantage of the OFLN over FLN, pi-sigma and GMDH, i.e. as percentage change in MSE is zero (fig. 5.2) then minimal number of 4 basis functions of up to degree 4 are sufficient for approximation for MSE = 0.08. Also, from (3), for N=2, D=4, L would be 15 but 6 of them are linearly dependent and eliminated during training. Hence, unlike GMDH and pi-sigma networks there are no repeated and redundant terms for higher degree representation. Results show that this property scales extremely well for systems with large numbers of inputs and outputs.

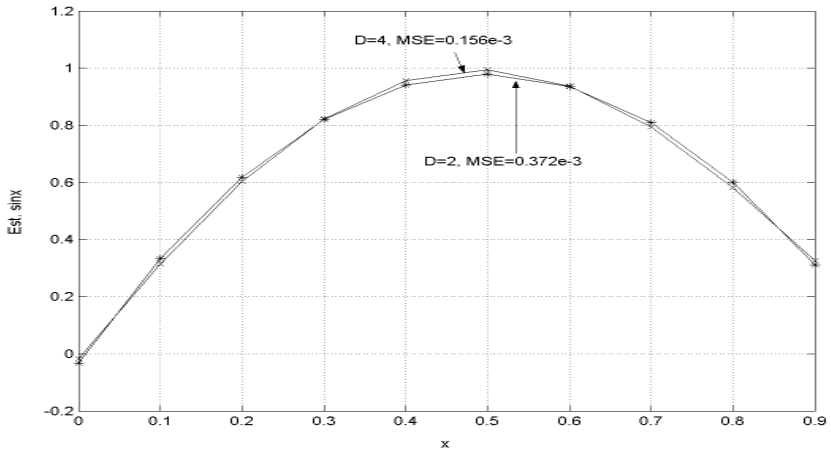

Fig. 5.1. sin(x) function approximation by OFLN for D=2 and D=4

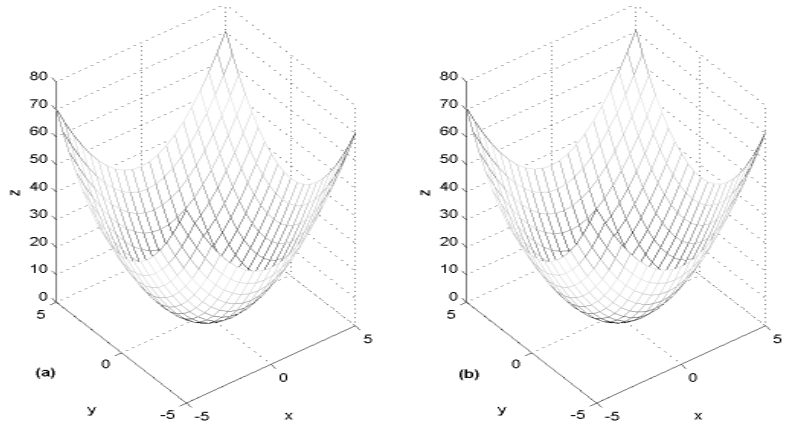

Fig. 5.2. (a) Modified Rastrigin function and (b) approximated by OFLN, D=4

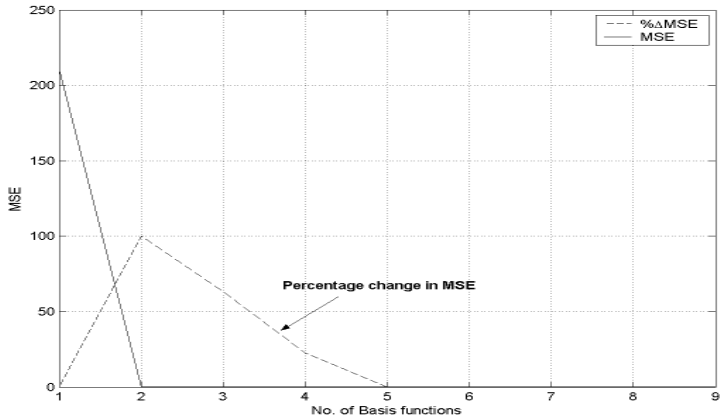

Fig. 5.3. MSE vs. No. of Basis Functions and percentage change in MSE for appx. Rastrigin function

5.2 Supervised Learning

Some examples for supervised learning are demonstrated for the OFLN. For comparison, a GMDH network [46][47] is designed using the Forward Prediction Error (FPE) criterion. In examples 5.2.1 and 5.2.2, an MLP is trained with back-propagation and the Levenberg-Marquardt algorithm. In examples 5.2.3 and 5.2.4 the MLP is trained using Output-Weight-Optimization Hidden-Weight-Optimization (OWO-HWO)[48]. The validation error is the MSE on some unseen patterns and is averaged for 3 sets of random data with ratio of 7:3 for training and validation.

5.2.1. California Housing

The first example for supervised learning is a benchmark function approximation problem "California Housing" from Statlib[49]. It has observations for predicting the price of houses in California. Information on the variables was collected using all the block groups in California from the 1990 Census. In this sample a block group on average includes 1425.5 individuals living in a geographically compact area. The geographical area included varies inversely with the population density. Distances among the centroids of each block group were computed as measured in latitude and longitude. All the block groups reporting zero entries for the independent and dependent variables were excluded. The final data contained 20,640 observations on 9 variables, which consists of 8 continuous inputs (median income, housing median age, total rooms, total bedrooms, population, households, latitude, and longitude) and one continuous output (median house value). The output is normalized by subtracting the mean and dividing by standard deviation for simplicity. The MSE obtained is 0.30, 0.38

and 0.35 for the OFLN, GMDH and MLP respectively. OFLN used 27 basis functions with D=4, GMDH is implemented with 4^{th} degree approximation, 27 PBF's over 20 iterations. An 8-18-1 MLP used a validation set for early stopping and converged at 31 epochs. Fig. 5.4 shows the training MSE (MSEt) and validation MSE (MSEv) vs. the number of basis function for OFLN with D=4. There is a need for applying the early stopping criterion here as the network tries to over fit the data by adding PBF's after 27 thereby resulting in the training MSE to further decrease but the validation error to increase. The 2 datasets are independent and it is required to obtain a generalized network, hence OFLN with 27 PBF is the desired network. Table 5.2 shows the MSE for various degree OFLN.

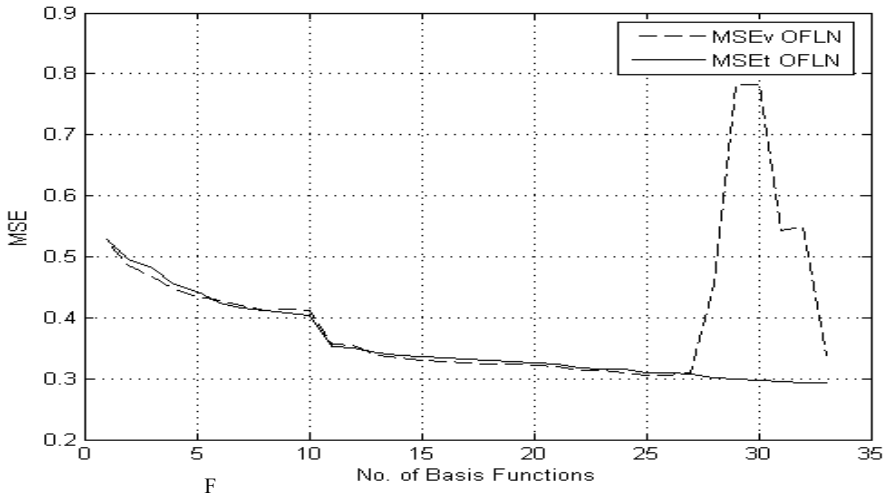

Fig. 5.4. Training and Validation MSE vs. No. of Basis Functions for California Housing approximation.

5.2.2. Inversion Technique for Radar Scattering

The second example comprises an empirical MIMO geophysical system for surface analysis from polarimetric radar measurements[50]. There are 20 inputs

corresponding to VV and HH polarization at L 30, 40 deg, C 10, 30, 40, 50, 60 deg, and X 30, 40, 50 deg and 3 outputs corresponding to rms surface height, surface correlation length, and volumetric soil moisture content in g/cubic cm. Fig. 5.5 shows the training and validation MSE vs. no. of basis function graph for ordered PBF's 1 to 60 for D=4. Results show good generalization capabilities for the OFLN.

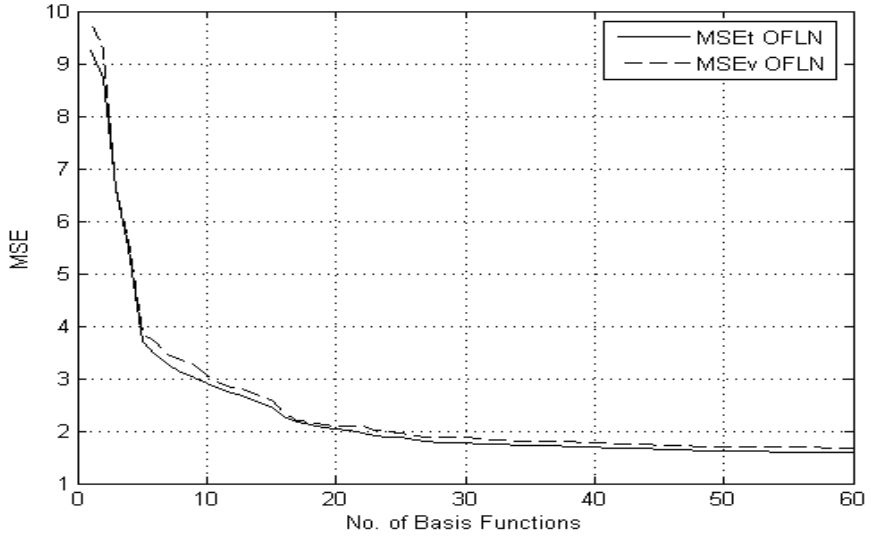

Fig. 5.5. Training and Validation MSE vs. No. of Basis Functions for Radar scatter dataset.

Table 5.1 gives the comparison for number of terms L used for system representation for FLN vs. OFLN, with L_{max} for higher degree set to 500 for the OFLN. Although L_{OFLN} for California housing example for D=4 is 377 for effective generalization only 27 basis functions are sufficient. Table 5.2 gives the corresponding training and validation MSE at each degree, compared with a Multi-layer Perceptron (20-6-3 for Radar over 100 epochs, 8-18-1 for Housing over 31 epochs). As can be seen for increase in D, MSE for OFLN decreases. The OFLN has the significant advantage that

as the network degree D increases, the training MSE decreases or at worst remains constant. This result follows from use of the Gram-Schmidt procedure, equation (44) where the training MSE is represented as non-increasing function with respect to the basis functions. Also for D^{th} degree learning, the training pattern file need be read only (D+1) times, the first time for normalizing the data set. These attributes make the OFLN computationally efficient over GMDH and similar PNN networks.

Table 5.1 No. of PBF's Comparison for OFLN vs. FLN

Degree	L_{FLN} Housing	L_{OFLN} Housing	L_{FLN} Radar	L_{OFLN} Radar
D=1	9	9	21	21
D=2	55	45	231	231
D=3	220	144	1771	284
D=4	715	337	10626	294

Table 5.2 Training and Validation MSE Comparison for OFLN vs. MLP

MSE	D=1	D=2	D=3	D=4	MLP
Training (Housing)	0.36	0.33	0.31	0.30	0.35
Validation (Housing)	0.37	0.33	0.32	0.31	0.35
Training (Radar)	3.69	1.52	1.38	1.36	1.43
Validation (Radar)	3.94	1.81	1.65	1.6	1.55

5.2.3. Noisy FM Demodulator

The third example is a parallel implementation of frequency discriminator-type Frequency Modulation (FM) demodulator that recovers a band-limited modulating signal from a frequency modulated signal distorted by additive colored noise of measured variance. If x[n] is the modulating signal, z[n] is the output of FM modulator with additive noise e[n], then for modulation index k_f, carrier amplitude A_c, carrier frequency f_c, modulating signal frequency f_m

$$z[n] = A_c \cdot \cos\left(2\pi f_c \cdot n + k_f \sum_{i=o}^{n} x[i] \right) + e[n] \tag{82}$$

$$k_f = \frac{2\pi f_c}{|\, x[n] \,|_{\max}} \tag{83}$$

1024 patterns are generated with z[i], $0 \leq i \leq 4$ as inputs and desired x[n] as output with values of A_c, f_c, f_m as 0.5, 0.1 and 0.1 respectively. Comparison of OFLN, MLP and GMDH based on training MSE (MSE_t) and validation MSE (MSE_v) vs. the number of basis functions is shown in fig. 5.6. OFLN gives a lower MSE for training and validation compared to MLP and GMDH. The number of basis functions for MLP under consideration is given by (Number of hidden units + N + 1). The GMDH network uses 5^{th} degree approximation for 50 iterations. Performance results for OFLN are comparatively better. Also, a system modeler can select a smaller size OFLN network with a trade-off in MSE, e.g. OFLN of size 40 compared to OFLN of size 60 has 2% additional training MSE at cost of 20 more PBF's.

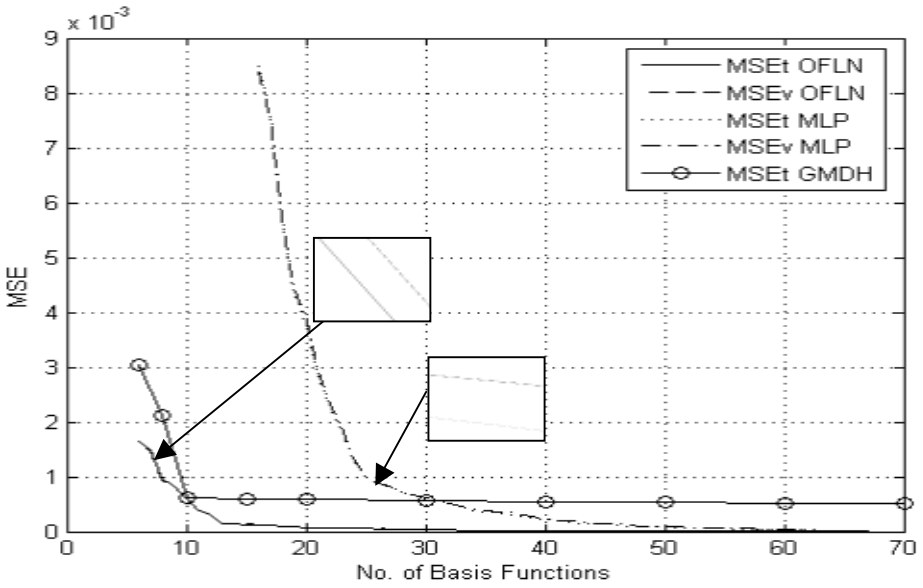

Fig. 5.6. Training and Validation MSE vs. vs. No. of Basis Functions comparison of OFLN, MLP and GMDH for FM Demodulator.

5.2.4. Matrix Inversion

Results for non-linear 2 by 2 matrix inversion problem are shown in fig. 5.7. The training file has 2000 patterns, each pattern consists of 4 input features and 4 output features. The input features, which are uniformly distributed between 0 and 1, represent a matrix and the four output features are elements of the corresponding inverse matrix. The determinants of the input matrices are constrained to be between .3 and 2. FLN, OFLN and MLP networks are compared in fig. 8. Note that as compared to the OFLN, FLN points are widely separated, giving the user few options as to network size. Also, from the figure we see that all three networks perform similarly when the number of basis functions is 23 or less. However, for this dataset, the MLP has an advantage for 24 or more basis functions.

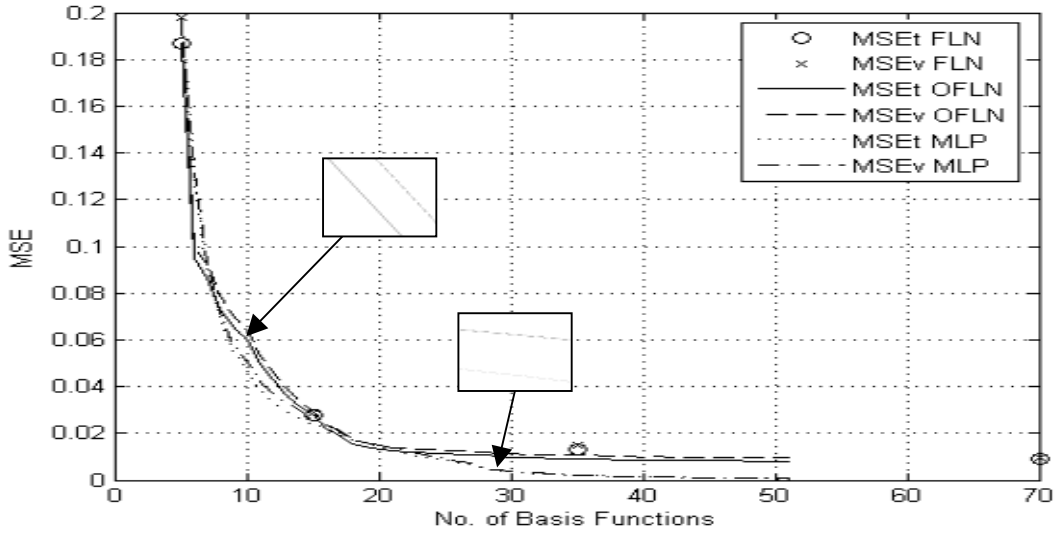

Fig. 5.7. Training and Validation MSE vs. No. of Basis Functions comparison of FLN, OFLN and MLP for 2x2 Matrix inversion.

5.3 Financial Market Forecasting

Success of Neural Networks in financial forecasting models is well-established [51-54] and use of OFLN for similar task is proposed here. Due to globalization, the global markets of pan America, Europe and the far-East are increasingly interdependent. It is often believed there exists a linear relationship between them [55]. Using OFLN a more appropriate relationship can be evaluated. OFLN is trained for Open Value data of 8 key indices, viz. NASDAQ (US), DJIA (US), Bovespa (Brazil), FTSE (UK), BSE (India), Hang-Seng (Hong Kong), KOSPI (South Korea) and NIKKEI (Japan). For demonstration non-linear relationships between KOSPI and other indices is presented below. The data collected ranges from January 2001 to January 2007. Samples are pre-processed for adequate representation and randomly distributed in sets for training, validation and processing sets. The training and validation curve in figure (5.8) show that an adequate selection of 80 basis function would give an optimal

network under given criterion. Using this network the open-value of KOSPI are predicted for a period of more than a year as shown in figure (5.9)

A normalized version of the MSE can be represented as Mean Absolute Percentage Error (MAPE) given by

$$MAPE = 100 \times \left(\frac{1}{M \cdot N_v} \sum_{i=1}^{M} \sum_{p=1}^{N_v} \frac{| y_{ip} - \overline{y}_{ip} |}{| y_{ip} |} \right) \tag{84}$$

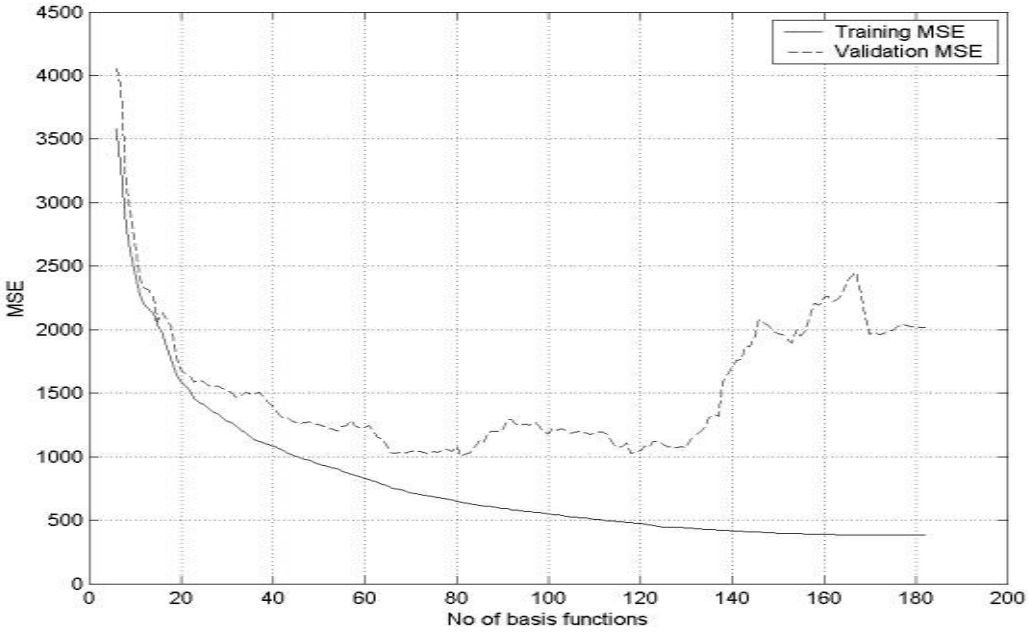

Fig. 5.8. Training and Validation MSE vs. No. of Basis Functions for OFLN, D=4 for KOSPI index.

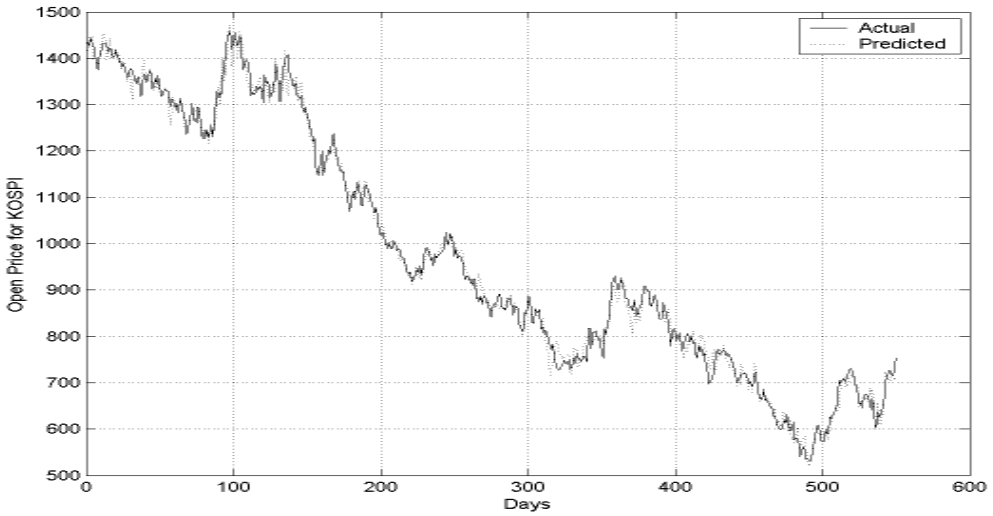

Fig. 5.9. Actual vs. Predicted open value for KOSPI index
as non-linear function of indices of global markets

Table (5.3) shows the comparison statistics for MAPE for OFLN (4^{th} order, 80 basis functions), MLP (7-90-1 using OWO – HWO algorithm) and GMDH (4^{th} order, 50 terms).

Table 5.3 MAPE Comparison for OFLN, MLP and GMDH Networks

Network Type	MAPE (%)	MAPE (σ)
OFLN	1.53	0.0144
MLP	1.57	0.0183
GMDH	4.19	0.12

The cumulative distribution function (CDF) of the MAPE presents a better insight into interpretation of MSE / MAPE for different networks. In figure (5.10) the CDF for MAPE is plotted, where the y-axis denotes the probability that the MAPE will be less

than a particular value for the network under consideration. Clearly, the more the probability for lesser MAPE the better is the estimator.

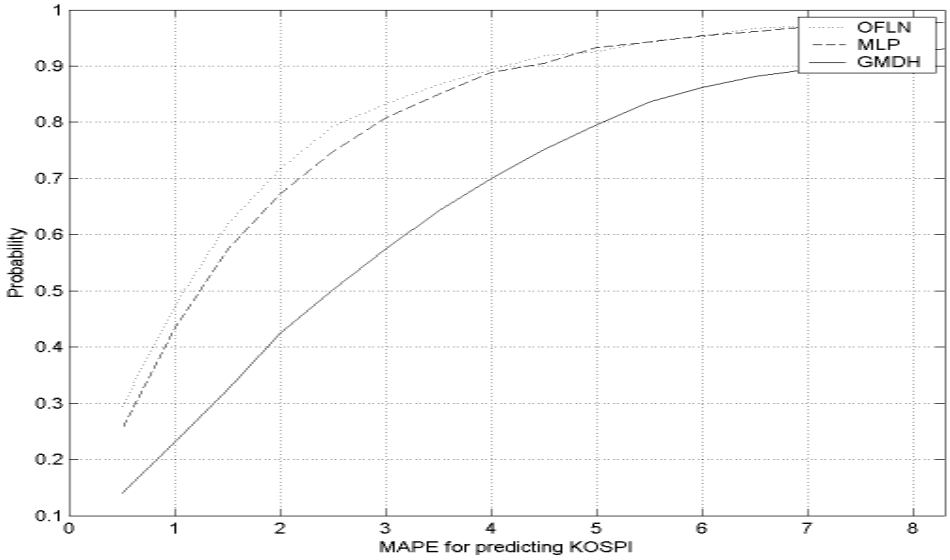

Fig. 5.10. MAPE CDF for OFLN, MLP and GMDH for predicting KOSPI index.

Using OFLN an equation form interpretation of the relationship between of the approximated function is possible. Let (x_1 – x_7 denote DJIA, NASDAQ, Bovespa, FTSE, BSE, Hang-Seng, NIKKEI) then the relationship defined for first few PBF is given by

$$971.78 + 230.32\cdot x_5 - 731.84\cdot x_5^3 + 194.08\cdot x_5\cdot x_2 + 586\cdot x_6\cdot x_1 + 133.34x_1\cdot x_5\cdot x_6 - 461.69\cdot x_1\cdot x_4\cdot x_5 + 78.166\cdot x_6\cdot x_6 + 667\cdot x_4^2\cdot x_5 + 35.852\cdot x_4\cdot x_5\cdot x_6 - 134.06\cdot x_4\cdot x_5\cdot x_6^2 - 48.72\cdot x_1\cdot x_4\cdot x_5^2 - 197.34\cdot x_2\cdot x_4\cdot x_5^2 + 188.96\cdot x_7 + 631.67x_4\cdot x_5\cdot x_5 + 149.61\cdot x_4 + 172.46\cdot x_2\cdot x_2\cdot x_5^2 + 48.827\cdot x_1\cdot x_2\cdot x_5 - 153.33\cdot x_2\cdot x_2\cdot x_5^2 - 104.96\cdot x_6 \quad (85)$$

which is an important finding as it suggests high dependency of the KOSPI's index on the BSE index, interestingly the correlation coefficient between the 2 is highest at 0.95.

5.4 OFLN Classifier Simulation

Performance of OFLN as classifier is compared with other classifiers like the MLP classifier, PLN classifier (PLNC) and Nearest Neighbor classifier (NNC). As in approximation, the network is trained by incrementing the degree D and OR iterations (N_{it}) and the network size selection is based on generalization rule, i.e. when the validation classification error for network starts increasing from a low. The MLP uses 100 iterations of OWO – HWO algorithm for learning and then OR for reducing the classification error (P_e). To increase the likelihood of good generalization cross-validation is performed where training and validation datasets are exchanged and the resulting values of P_e are then averaged.

5.4.1. Geometric Shape Recognition

The geometric shape recognition problem is to classify four geometric shapes: ellipse, triangle, quadrilateral, and pentagon [56]. Each shape consists of a matrix of size 64 x 64. For each shape, 200 training patterns are generated using different degrees of deformation. The deformations included rotation, scaling, translation, and oblique distortions. The feature set is ring-wedge energy (RNG), and has 16 features and form the input set **x**.

Table 5.4 gives a comparison of training P_e (P_{et}) and validation P_e (P_{ev}) for OFLN of various degrees D and various OR iterations N_{it}. It is seen that as the network degree increases P_{et} goes on decreasing as increase in D results in training MSE decrease and OR works effectively to decrease the error. However from D=2 to D=3,4 and for increase in N_{it} the corresponding validation error P_{ev} increases which indicates that the network tends to become over-specialized and a good stopping criterion is thus for D=2 and N_{it} = 5. The 16-136-1 MLP gives a P_{et} = 0.34 and P_{ev} = 4.94.

Table 5.4 Training and Validation Classification Error for OFLN

OFLN Network	N_{it} = 5		N_{it} = 10		L
	P_{et}	P_{ev}	P_{et}	P_{ev}	
D=1	12.84	13.76	11.93	13.77	17
D=2	1.44	4.85	0.36	6.47	153
D=3	0	12.55	0	15.38	499
D=4	0	9.31	0	10.93	499

Figure 5.11 shows the graph for training and validation classification error for OFLN and MLP vs. number of basis functions. A fully-connected 16-170-4 MLP using OWO – HWO for 100 iterations is trained. The number of basis functions for a fully-connected feed-forward MLP is as defined before (Number of inputs + Number of hidden units + 1). OFLN selected is of degree 2 starting with L=171, but 18 of them are linearly dependent and eliminated and that gives a 2^{nd} degree OFLN with 153 PBF's. P_{et}

for MLP is a decreasing function for the dataset under consideration. All the basis functions are required for the OFLN to achieve near similar performance to MLP, this is because after the OR algorithm the PBF's are not re-ordered again for the OFLN whereas they are for the MLP. The computation time for OFLN is less than for the MLP.

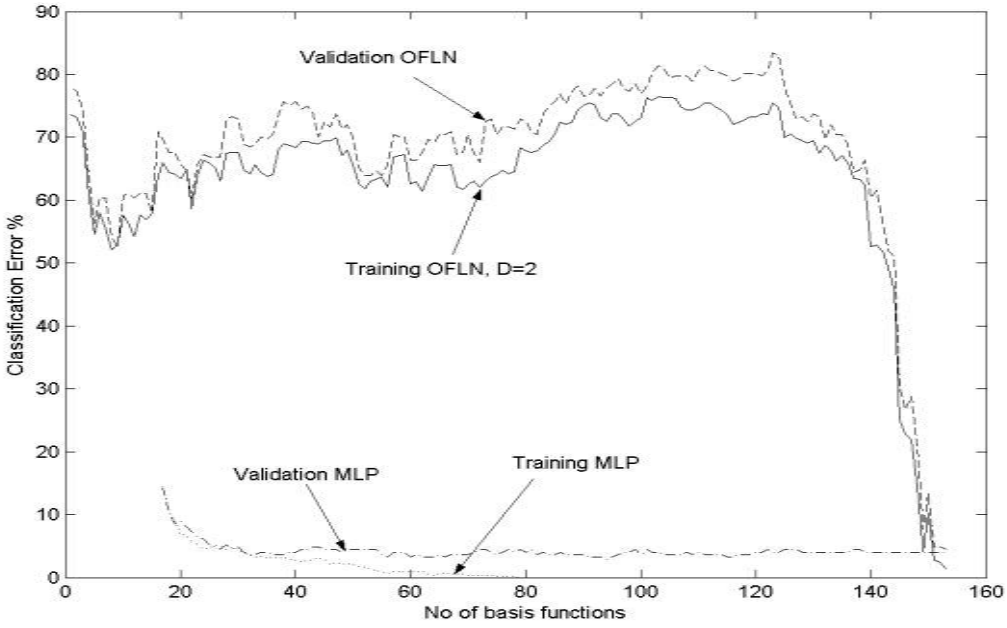

Fig. 5.11 Training and Validation P_e comparison for OFLN and MLP for shape recognition.

5.4.2. Hand-Printed Numeral Dataset Recognition

The raw data for hand-printed numeral recognition consists of images from hand printed numerals collected from 3,000 people by the Internal Revenue Service (IRS)[57]. 300 characters are randomly chosen from each class to generate 3000 character dataset. Images are 32 by 24 binary matrices. An image-scaling algorithm is

used to remove size variation in characters. The feature set contains 16 elements. The 10 classes in this problem correspond to the 10 Arabic numerals.

Fig 5.12 shows a comparison of OFLN of degree 2 with 153 PBF's and N_{it} = 3 and 16–136 –10 fully-connected MLP using OWO – HWO for 60 iterations is trained. Both the networks show near similar error performance for number of basis functions greater than 137. The OFLN for D=4 gives a better P_{et} over MLP but is not shown in figure as the P_{ev} is higher than for D=2 case. In this case from the figure P_{ev} is 8.94 and 8.52 for OFLN and MLP respectively. Table 5.5 compares the error for OFLN, MLP, PLNC and NNC. PLNC used a maximum of 153 clusters and NNC used 115 clusters. OFLN gives better performance over the PLNC[38][58] and NNC[56] in this case.

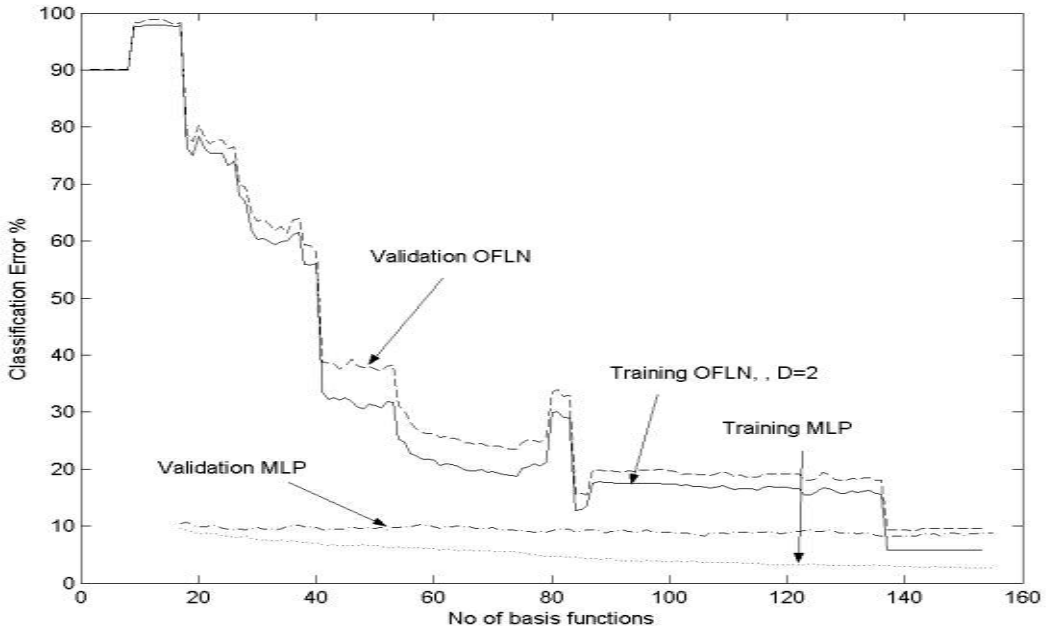

Fig. 5.12 Training and Validation P_e comparison for OFLN and MLP for numeral recognition.

Table 5.5 Comparison of Classification Error for Arabic numeral recognition for OFLN, MLP, PLNC, NNC

Network Type	P_{et}	P_{ev}
OFLN	4.04	8.94
MLP	2.64	8.52
PLNC	4.18	10.14
NNC	5.42	9.16

5.4.3 Image Segmentation

In this example the training and validation datasets are generated from segmented images of land-types. Each segmented region is a separate histogram equalized to 20 levels. The joint probability density of pairs of pixels separated by a given distance and a given direction is estimated using 0, 90, 180, 270 degrees for the directions and 1, 3, and 5 pixels for the separations. The density estimates are computed for each classification window. For each separation, the co-occurrences for for the four directions are folded together to form a triangular matrix. From each of the resulting three matrices, six features are computed: angular second moment, contrast, entropy, correlation, and the sums of the main diagonal and the first off diagonal. This results in 18 features for each classification window. Four regions of land use/cover types are identified in the images per Level I of the US Geological Survey Land Use/Land Cover Classification System: urban areas, fields or open grassy land, trees (forested land), and water (lakes or rivers) [59]. Thus for this problem $N = 18$, $N_c = 4$.

A D=3 OFLN is trained with N_{it} = 4. The training and validation classification error graph is shown in figure 5.13. A 18-101-4 fully-connected feed-forward MLP using OWO-HWO is trained and pruned and the corresponding error plot it shown in figure 5.14.

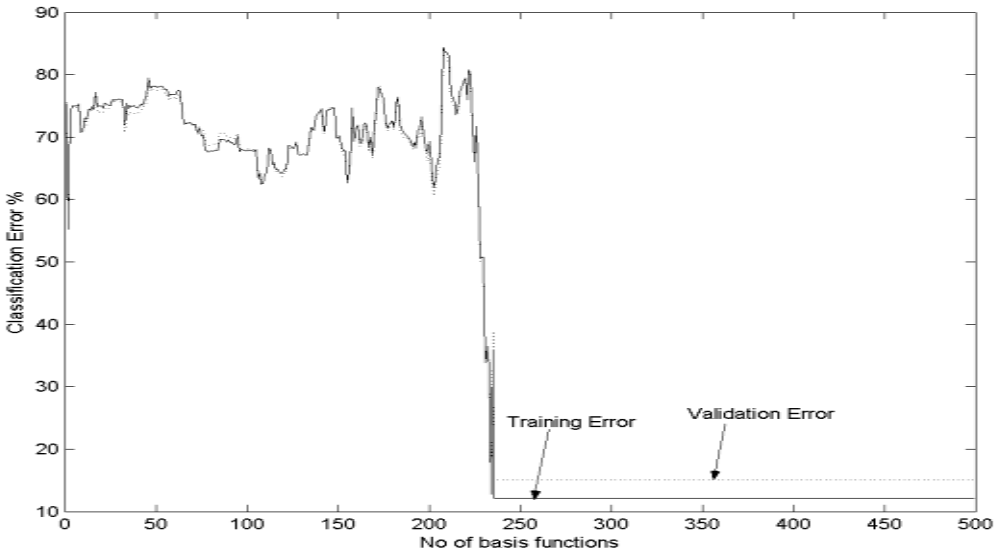

Fig. 5.13 Training and Validation P_e for OFLN, D=3 for image segmentation.

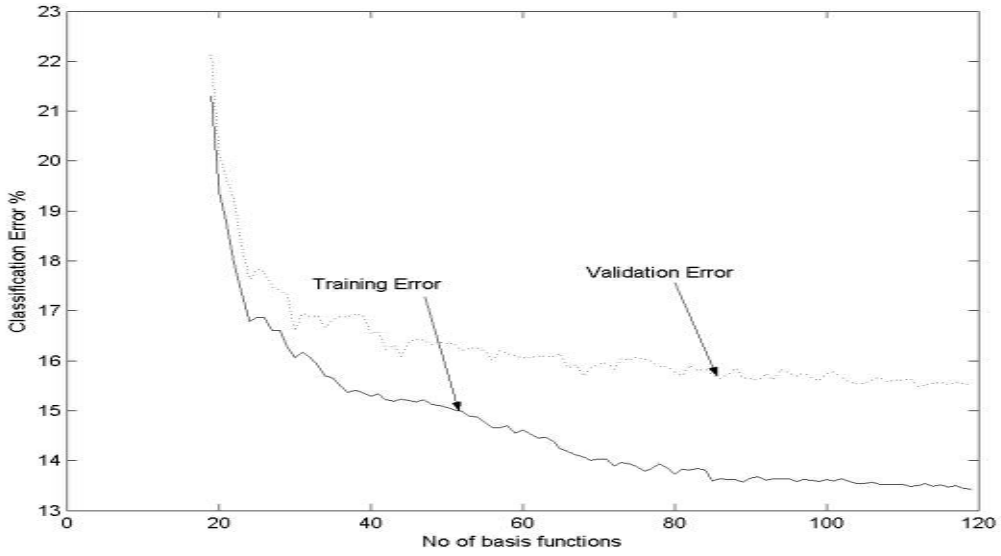

Fig. 5.14 Training and Validation P_e for MLP for image segmentation.

As can be seen, P_{et} and P_{ev} corresponding to OFLN and MLP for 236 and 120 basis functions are 12.1% and 15.1% for OFLN with 236 basis functions and 13.4% and 15.5% for MLP with 120 basis functions. For the segmentation problem, the OFLN provides a better classification error compared to the MLP.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this thesis the OFLN is proposed as a valuable network for approximation and classsification. It is seen that the OFLN gives a concise, structured, easily interpretable, methodically ordered and computationally efficient network for supervised, non-parametric MIMO systems. It improves on the FLN and a few other PNN networks.

An efficient method for generating, orthonormalizing, pruning and ordering higher order polynomial basis functions funcions together constitute the OFLN. The elimination of the linearly dependent functions and other less useful basis functions is important in containing the combinatorial explosion which in turn allows better approximation capabilities. The number of passes through the training data is reduced using higher order correlations. A set of different order and size complexity polynomial networks based on incremental size of polynomial basis functions which explicilty explain the trade-off between performance and network size is presented to the modeler using the OFLN. As demonstrated the OFLN can certainly be applied to many nonlinear function approximation, structure identification and optimization and classificaiton problems. It is a however a known fact that there is a “no one-network-fit-all” solution in Neural Networks.

There is enough scope with suitable rewards for future work in the OFLN.

1. The OFLN algorithm can be extended for a cluster or distributed computing architecture to suit more complex number crunching problems like the weather forecasting, space exploration and natural disaster forewarning systems.
2. With a few modificaitons in algorithmic implementation, the OFLN can be extended to time-series modeling problems. For time-series problems a highly useful and complex OFLN can grow in multiple dimensions of order, lag-window size and estimate output-window.
3. Depending on the application, the stopping criterion for OFLN approximation and classification can be automated by keeping a check on the training and validaiton error during the incremental growth of network, however it would increase computational complexity.
4. The performane complexity of Gram-Schmidt orthonormalization procedure and few other techniques like the Householder transformation, Givens method, Singular Value Decompostion can be compared.

APPENDIX A

REGRESSION-BASED NETWORK APPROXIMATION TO BAYESIAN DISCRIMINANT FUNCTION

Consider a classification network with the input feature **x** and output discriminant function $\mathbf{y}_i(\mathbf{x})$ for N_v for i between 1 and N_v. Let $N_v(i)$ the number of patterns belonging to class i, S(i) the set of patterns that correspond to class i, $f_x(\mathbf{x})$ denote the probability density of feature vector **x**, $f(\mathbf{x}|i)$ the conditional density of feature vector **x** given it belongs to class i and P(i) the probability that feature vector comes from class i.

The optimal Bayesian discriminant $d_i(\mathbf{x})$ is given as

$$d_i(\mathbf{x}) = P(i|\mathbf{x}) \tag{A1}$$

where $P(i|\mathbf{x})$ is the aposteriori probability that given a vector **x** it belongs to class i. Let the desired output y_{ip} for correct class j defined as

$$y_{ip} = \delta(i-j) \tag{A2}$$

and the estimated output for p^{th} pattern by the regression based classifier be denoted by $\overline{y}_{ip}$. Then the MSE for the network given N_v training patterns is given by

$$E_t = \frac{1}{N_v}\sum_{p=1}^{N_v}\sum_{i=1}^{N_c}\left(y_{ip}(\mathbf{x}) - \overline{y}_{ip}(\mathbf{x})\right)^2 \tag{A3}$$

The expected squared error between network output $\overline{\mathbf{y}}_i$ and optimal Bayes discriminant is given by

$$E_B = \sum_{i=1}^{N_c} E\left[\left(d_i(\mathbf{x}) - \overline{y}_i(\mathbf{x})\right)^2\right] \tag{A4}$$

Theorem[44]: As the training patterns N_v increase, E_t approaches the (E_B + K), where K is a constant.

Proof: Given the definition of t_{ip}, equation (A3) is rewritten as

$$E_t = \sum_{i=1}^{N_c} \sum_{p \in S(i)} \frac{1}{N_v} \left[\left(1 - \overline{y}_i(\mathbf{x})\right)^2 + \sum_{j=1, j \neq i}^{N_c} \overline{y}_j{}^2(\mathbf{x}) \right] \tag{A5}$$

$$E_t = \sum_{i=1}^{N_c} \frac{N_v(i)}{N_v} \left[\frac{1}{N_v(i)} \sum_{p \in S(i)} \left[\left(1 - \overline{y}_i(\mathbf{x})\right)^2 + \sum_{j=1, j \neq i}^{N_c} \overline{y}_j{}^2(\mathbf{x}) \right] \right] \tag{A6}$$

As $N_v \to \infty$, $N_v(i) / Nv \to P(i)$ and (A6) is written as

$$E_t = \sum_{i=1}^{N_c} P(i) \left[\int \left(\left(1 - \overline{y}_i(\mathbf{x})\right)^2 + \sum_{j=1, j \neq i}^{N_c} \overline{y}_j{}^2(\mathbf{x}) \right) \mathrm{f}(\mathbf{x} \mid i) d\mathbf{x} \right] \tag{A7}$$

$$= \sum_{i=1}^{N_c} \left[\int \left(1 - \overline{y}_i(\mathbf{x})\right)^2 \mathrm{f}(\mathbf{x} \mid i) P(i) d\mathbf{x} + \int \sum_{j=1, j \neq i}^{N_c} \overline{y}_j{}^2(\mathbf{x}) \mathrm{f}(\mathbf{x} \mid i) P(i) d\mathbf{x} \right] \tag{A8}$$

$$= \sum_{i=1}^{N_c} \left[\int \left(1 - \overline{y}_i(\mathbf{x})\right)^2 \mathrm{f}(\mathbf{x} \mid i) P(i) d\mathbf{x} + \int \sum_{j=1, j \neq i}^{N_c} \overline{y}_i{}^2(\mathbf{x}) \mathrm{f}(\mathbf{x} \mid j) P(j) d\mathbf{x} \right] \tag{A9}$$

$$= \sum_{i=1}^{N_c} \left[\int \left(1 - 2 \cdot \overline{y}_i(\mathbf{x})\right) \mathrm{f}(\mathbf{x} \mid i) P(i) d\mathbf{x} + \int \overline{y}_i{}^2(\mathbf{x}) \sum_{j=1}^{N_c} \mathrm{f}(\mathbf{x} \mid j) P(j) d\mathbf{x} \right] \tag{A10}$$

$$= \sum_{i=1}^{N_c} \left[\int \left(1 - 2 \cdot \overline{y}_i(\mathbf{x})\right) P(i \mid \mathbf{x}) \mathrm{f}(\mathbf{x}) d\mathbf{x} + \int \overline{y}_i{}^2(\mathbf{x}) \mathrm{f}(\mathbf{x}) d\mathbf{x} \right] \tag{A11}$$

where the following substitutions are from Bayesian theory

1. $\mathrm{f}(\mathbf{x} \mid i) P(i) = \mathrm{f}(\mathbf{x}, i) = P(i \mid \mathbf{x}) \mathrm{f}(\mathbf{x})$ and (A12)

2. $\sum_{j=1}^{N_c} \mathrm{f}(\mathbf{x} \mid j) P(j) = \mathrm{f}(\mathbf{x})$ (A13)

Then (85) is,

$$E_t = \sum_{i=1}^{N_c} \left[\int \left[\left(1 - 2 \cdot \overline{y}_i(\mathbf{x})\right) d_i(\mathbf{x}) + \overline{y}_i^{\,2}(\mathbf{x}) \right] \mathrm{f}(\mathbf{x}) d\mathbf{x} \right] \tag{A14}$$

$$E_t = \sum_{i=1}^{N_c} \left[\int \left[\left(\overline{y}_i(\mathbf{x}) - d_i(\mathbf{x}) \right)^2 \mathrm{f}(\mathbf{x}) d\mathbf{x} \right] + \int \left(1 - d_i(\mathbf{x})\right) d_i(\mathbf{x}) \mathrm{f}(\mathbf{x}) d\mathbf{x} \right] \tag{A15}$$

$$E_t = \sum_{i=1}^{N_c} E\left[\left(\overline{y}_i(\mathbf{x}) - d_i(\mathbf{x}) \right)^2 \right] + \sum_{i=1}^{N_c} \left[\int \left(1 - d_i(\mathbf{x})\right) d_i(\mathbf{x}) \mathrm{f}(\mathbf{x}) d\mathbf{x} \right] \tag{A16}$$

The second term in (A16) is independent of any network parameters and is thus a constant K. Thus $E_t = E_B + K$, it implies that network with estimated output $\overline{y}$, which minimizes the MSE yields the optimal Bayes discriminant function in the minimum mean squared error sense. *Q.E.D.*

REFERENCES

[1] Warren McCulloch and Walter Pitts, “A Logical Calculus of Ideas Immanent in Nervous Activity”, *Bulletin of Mathematical Biophysics*,1943, 5:115-133.

[2] A. G. Ivakhnenko, “Heuristic self-organization in problems of engineering cybernetics”, *Automatica*, Vo1.6, No.2, 1970, pp.207-219.

[3] S. J. Farlow ed., *Self-organizing Methods in Modeling, GMDH-type Algorithms*, Marcel Dekker, Inc., New York, 1984.

[4] Simon Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall PTR, 1998.

[5] Christopher M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[6] Gao, X.M.; Gao, X.Z.; Tanskanen, J.M.A.; Ovaska, S.J., "Power prediction in mobile communication systems using an optimal neural-network structure", *IEEE Trans. Neural Networks*, Vol. 8, Iss. 6, 1997 pp.1446 - 1455.

[7] Fulcher, G.E.; Brown, D.E., "A polynomial network for predicting temperature distributions", *IEEE Trans. Neural Networks*, Vol. 5, Iss. 3, 1994, pp.372 - 379.

[8] Lajbcygier, P., "Improving option pricing with the product constrained hybrid neural network", *IEEE Trans. Neural Networks*, Vol. 15, Iss. 2, 2004, pp.465 - 476.

[9] M. H. Stone, "The Generalized Weierstrass Approximation Theorem", *Mathematics Magazine* 21 (5), 1958, pp.237–254.

[10] Minsky, M. and Papert, S., *Perceptrons*, MIT Press, Cambridge,1969.

[11] P. Werbos, “Beyond regression: New tools for prediction and analysis in the behavioral sciences,” Ph.D. dissertation, *Committee on Appl. Math*., Harvard Univ., Cambridge, MA, Nov.1974.

[12] K. Fukushima, "Cognitron: A Self-Organizing Multilayered Neural Network", *Biological Cybernetics*, Vol. 20, No. 3/4, NOV. 1975, pp. 121-136.

[13] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. N&l. Acud. Sci*., vol. 81, pp. 3088-3092, 1984.

[14] T. Kohonen, *Associative Memory: A System-Theoretical Approach*, Springer-Verlag, Berlin, 1977.

[15] S. Grossberg, "Adaptive Pattern Classification and Universal Recoding, I: Parallel Development and Coding of Neural Feature Detectors", *Biological Cybernetics*, Vol. 23, 1976, pp.121-134.

[16] Y.H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley Pub, 1989.

[17] W. Kolmogorov, “Interpolation and extrapolation of stationary series”, *Bulletin de Academie de*, U.S.S.R.,2:3, 1942.

[18] J. Macias.A., A. Sierra., F. Corbacho, "Evolving and assembling functional link networks", *IEEE Trans. Evolutionary Computation*, Vol. 5, No. 1, 2001, pp. 54-65.

[19] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, MA: Addison-Wesley, 1989.

[20] E.J. Hartman, J.D. Keeler, J.M. Kowalski, "Layered Neural Networks with Gaussian Hidden Units as Universal Approximations," *Neural Computation*, vol. 2, No. 2, 1990, pp. 210-215.

[21] Michael T. Manry, Steven J. Apollo, and Qiang Yu, "Minimum Mean Square Estimation and Neural Networks," *Neurocomputing*, vol. 13, September 1996, pp. 59-74.

[22] T. M. Jelonek and James P. Reilly, "Maximum Likelihood Estimation for Direction of Arrival Using a Nonlinear Optimizing Neural Network," *IJCNN 1990*, vol. I, pp 253-258.

[23] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill Book Company, New York, 1965.

[24] R. Battiti, "First- and second-order methods for learning: Between steepest descent and Newton's method," *Neural Computation*,1992, vol. 4, pp.141-166.

[25] D. G. Luenherger, *Linear and Nonlinear Programming*, 2nd ed., MA: Addison-Wesley, 1989.

[26] C. Bishop, "Exact calculation of the Hessian matrix for the multilayer perceptron," *Neural Computation*, vol. 4, 1992, pp. 494-501.

[27] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, 1986, vol. 323, pp. 533-536.

[28] M. Klassen, Y. H. Pao., V. Chen, "Characteristics of the functional link net: a higher order delta rule net", *IEEE Con. Neural Networks*, Vol. 1, 1988, pp. 507-513.

[29] Moody, J.E., Hanson, S.J., and Lippmann, R.P.,"The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems", *Adv. Neural Inf. Proc. Sys.* Vol. 4, pp. 847-854.

[30] Weiss, S.M. & Kulikowski, C.A., *Computer Systems That Learn*, Morgan Kaufmann, 1991.

[31] Shin, Y.; Ghosh, J., "The pi-sigma network: an efficient higher-order neural network for pattern classification and function approximation Neural Networks", *Proc. of IJCNN 1991*, Vol. 1, pp.13 - 18.

[32] Shin, Y.; Ghosh, J., "Ridge polynomial networks", *IEEE Trans. Neural Networks*, Vol. 6, Iss. 3, 1995, pp. 610 - 622.

[33] J. Macias.A., A. Sierra., F. Corbacho, "Evolving and assembling functional link networks", *IEEE Trans. Evolutionary Computation*, Vol. 5, No. 1, 2001, pp. 54-65.

[34] Gilbert Strang, *Introduction To Linear Algebra*, Wesley-Cambridge Press, 1993

[35] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, 1991, pp. 302–309.

[36] W. Kaminski and P. Strumillo, "Kernel Orthonormalization in Radial Basis Function Neural Networks," *IEEE Trans. Neural Networks*, Vol. 8, No. 5, 1997, pp. 1177-1183.

[37] F. J. Maldonado, M. T. Manry, T. Kim, "Finding optimal neural network basis function subsets using the Schmidt procedure", *Proc. of IJCNN*, Vol. 1, 2003, pp. 444 - 449.

[38] J. Li, M. T. Manry, P. Narasimha, C. Yu, "Feature Selection Using a Piecewise Linear Network", *IEEE Trans. Neural Networks*, Vol 17, No. 5, 2006, pp.1101-1115.

[39] R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, 2nd Ed,Wiley Interscience, 2000.

[40] K. Fununaga, *Statistical Pattern Recognition*, 2nd Ed., Academic Press, NY, 1990.

[41] M.D. Srinath, P.K. Rajasekaran, *An Introduction to Statistical Signal Processing With Applications*, John Wiley and Sons, 1979.

[42] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.

[43] C.K.I. Williams and D. Barber, “Bayesian Classification with Gaussian Processes,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, 1998, pp. 1342-1351.

[44] D.W. Ruck, S. Rogers, M. Kabrisky, M. Oxley, B. Suter, "The multilayer perceptron as an approximation to a bayes optimal discriminant function", *IEEE Trans. Neural Networks*, 1990, pp. 296-298.

[45] R. G. Gore, J. Li, M. T. Manry, L. M. Liu, C. Yu and J. Wei, "Iterative Design of Neural Network Classifiers Through Regression", *Int. Journal Artificial Intelligence Tools*, Vol 14, Issues 1&2, 2005.

[46] Tetko, I. V., Gasteiger, J., Todeschini, R., Mauri, A., Livingstone, D., Ertl, P., Palyulin, V. A., Radchenko, E. V., Zefirov, N. S., Makarenko, A. S., Tanchuk, V. Y., Prokopenko, V. V. Virtual computational chemistry laboratory - design and description, *J. Comput. Aid. Mol.*, Dec. 2005, 19, pp. 453-63.

[47] VCCLAB, Virtual Computational Chemistry Laboratory, 2005, *Available: http://www.vcclab.org.*

[48] Changhua Yu, Manry, M.T., "A modified hidden weight optimization algorithm for feedforward neural networks", *IEEE Asilomar Con. on Signals, Systems and Computers*, 2002. Vol. 2, 2002, pp.1034-1038

[49] R.K. Pace, R. Barry, “Sparse spatial autoregressions”, *Statistics & Probability Letters*, Vol. 33, No. 3, May 1997, pp. 291-297.

[50] Y. Oh, K. Sarabandi, and F.T. Ulaby, "An Empirical Model and an Inversion Technique for Radar Scattering from Bare Soil Surfaces" , *IEEE Trans. Geoscience and Remote Sensing*, 1992, pp. 370-381.

[51] Buscema M., & Sacco P. L., "Feedforward networks in financial predictions: the future that modifies the present", *Expert Systems*, v. 17, No3, 2000, pp. 149-169.

[52] Donaldson G. R. & Kamstra M., (1996). Forecast combining with neural networks, *Journal of Forecasting*, v. 15, pp. 49-61.

[53] White H., "Economic prediction using neural networks: the case of IBM daily stock returns in neural networks", *Finance & Investing Ed.*, by Trippi R. R. & Turban E., pp. 315-328.

[54] Yoon Y. & Swales G., "Predicting Stock price performance: A neural networks approach in neural networks", *Finance & Investing Ed*., by Trippi R. R., & Turban E., pp. 315-328.

[55] A. N. Burgess, "Modeling Relationships between International Equity Markets using Computational Intelligence", *Knowledge-Based Intelligent Electronic Systems, Proc. KES* 1998, Vol. 3, pp.13 - 22.

[56] H. C. Yau, M. T. Manry, "Iterative Improvement of a Nearest Neighbor Classifier", *Neural Networks*, Vol. 4, 1991, pp. 517-524.

[57] W. Gong, H. C. Yau, and M. T. Manry, "Non-Gaussian Feature Analyses Using a Neural Network," *Progress in Neural Networks*, vol. 2, 1994, pp. 253-269.

[58] A. A. Abdurrab, M. Manry, J. Li, S. Malalur, R. Gore, "A Piecewise Linear Network Classifier", *IEEE Int. Joint Conference Neural Networks,* 2007, (accepted).

[59] R.R. Bailey, E. J. Pettit, R. T. Borochoff, M. T. Manry, and X. Jiang, "Automatic Recognition of USGS Land Use/Cover Categories Using Statistical and Neural Network Classifiers," *Proc. of SPIE OE/Aerospace and Remote Sensing*, 1993.

BIOGRAPHICAL INFORMATION

Saurabh Sureka received his BSEE degree from Amravati University, India in 2002 and is currently pursuing MSEE from University of Texas at Arlington, USA. He was involved in various projects in communication technologies with GTL Limited (Mumbai, India), Infosys Technologies (Bangalore, India) from 2002 to 2004 and with Qualcomm (San Diego, USA) in 2006.

He is currently with the Image Processing and Neural Network Laboratory (IPNNL), Arlington where he is working on developing neural network solutions to signal processing, communications and econometrics. His current research interests include the theory and application of Polynomial Neural Networks.

He is the recipient of university outstanding performers scholarships (1999-2002), Priyadarshani International Scholarship for Engineering (2000), Neural Network laboratory scholarships (2005-2006) and has served as the Founding Chairman of IEEE SSGMCE Shegaon Student Chapter (2001-02).