THE UNIVERSITY
of LIVERPOOL

MULTI LAYER PERCEPTRON AND CONIC SECTION FUNCTION NEURAL NETWORKS APPLIED TO BREAST CANCER RISK FACTORS INCLUDING ASYMMETRY

Thesis submitted in accordance with the requirements of the
University of Liverpool for the degree of Doctor of Philosophy

By

Min Soe Hane Aung

November 2003

Abstract

This thesis describes a software simulation based project in the area of Artificial Neural Networks, Image Processing and Breast Cancer. Breast Cancer predisposition data is used in training for the comparison of two types of feed forward neural network. The first being the Multi Layer Perceptron and the second being Conic Section Function Neural Networks, which is a hybrid network between Multi-layer Perceptrons and Radial Basis Function Neural Networks. This project also uses Digital Image Processing methods to extract asymmetry features from digitised mammogram pairs as part of the predisposition criteria. Standard Error Back Propagation is used to train the Mutli Layer Perceptron and a mixed Error Back Propagation and Orthogonal Least Squares Algorithm is used in Conic Section Function Network learning. This document also includes additional work on developing Conic Section Function Networks by Genetic Algorithms using the Iris Plant Data set.

The two forms of Neural Network are evaluated using the Log Likelihood Error Function by way of the N-Fold Cross Validation Method. Evaluation of classification results are compared using Receiver Operator Curves.

This work is based on the Mathworks MATLAB platform for algorithm simulation and RADWORKS imaging software for mammogram digitisation.

Findings in this study show that convergence in learning is more successful in the case of Multi-Layer Perceptrons however gaining successful classification required further study.

Acknowledgements

The author of this thesis would like to thank the following people for their invaluable advice and support during the course of this project.

Department of Electrical Engineering and Electronics:

Dr John Marsland

Professor Asoke Nandi

Dr Jason Ralph

Head of the Department of Medial Imaging:

Dr Diane Scutt

Consultant Oncologist Royal Liverpool University Hospital:

Mr Sunn Myint

Contents

List of Figures

List of Tables

Introduction

Artificial Neural Networks are applicable in a wide variety of problems such as prediction, pattern recognition and classification. Although the founding principles and the theories behind Artificial Neural Networks are not new, the advent of increasing computing power has seen the application of Neural Networks become widespread. Application of Neural Networks in the Medical field more specifically in cancer research [14][17][43][49] is recognised as a useful method, especially in difficult classifier problems where the data is highly variable and the search space has high dimensions.

One particular sub section of cancer research that has had much work done in relation to Neural Networks [40][105] is breast cancer research and in particular Mammography. Neural Networks are widely applied in radiographic studies and image pattern recognition as a whole. Many recognition networks have been developed especially in the detection of micro-calcifications (seen as small speckled clusters) in Mammograms which is a known pre-cursor to malignant bodies [17][105]. This study however takes a broader view of the breast cancer development problem and considers multiple pre-dispositions for cancer. This stance is a more difficult and unpredictable problem, cancer is known to have developed in seemingly healthy cases with no pre-dispositions. The use of Neural Networks to classify such grey problems does have an application here. Clinicians often have to inspect large numbers of Mammograms during mass screening programme. This has been known to lead to large backlogs and high risk cases being overlooked.

This project relates to a recent study carried out by *Scutt et al* [83] investigating asymmetry and its relation to cancer development. The first chapter in this thesis

describes this study and the use of left and right comparison practices in Mammography screening in greater detail. Chapter 2 introduces the principles behind Neural Networks and describes a standard Neural Network known as the Multi Layer Perceptron. Chapter 3 introduces a novel Neural Network known as the Conic Section Function network, this a highly complex network with increased generality which may be beneficial in this case. Chapter 4 describes standard Digital Image Processing methods applied to mammograms. Chapter 5 gives the results of the use of the Image processing methods and also describes methods specific to the Mammograms in this study. This chapter also relates to the asymmetry aspect and gives methods for extracting these features. Chapter 6 describes the gained asymmetry quantities and relates them to cancer and non-cancer cases, this chapter further interprets the data with other predisposition information and simulates Neural Networks giving results and evaluating discriminating ability of different configurations and the Multi Layer Perceptron and Conic Section Function Neural Network. Chapter 7 is additional work in relating to specifically Conic Section Function Networks only and the use of generalised network development method based on evolutionary principles. The final chapter 8 concludes the work done in this study it describes and reflects on the results gained in chapters 5, 6 and 7.

CHAPTER 1

Mammography and Cancer Risk Assessment.

1.1 Breast Cancer.

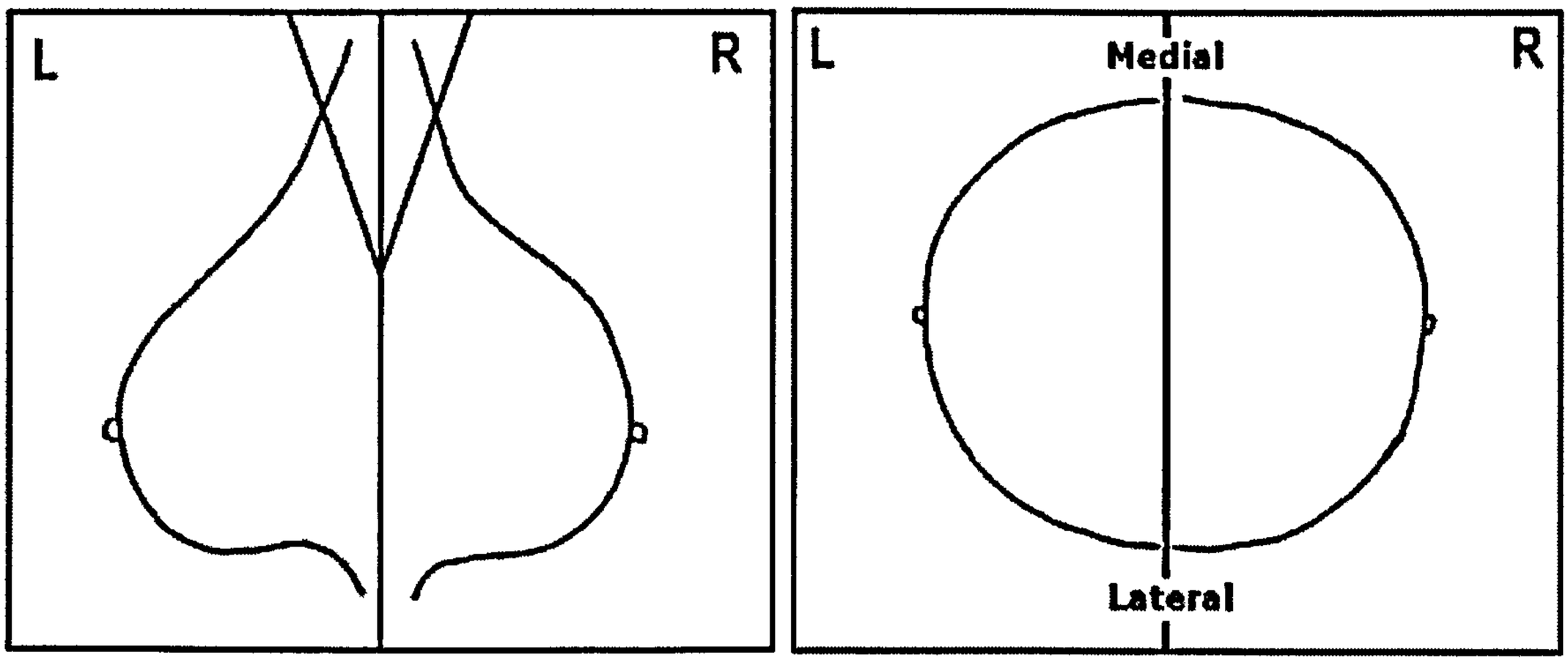
Breast cancer is among the most common and deadly of all cancers, occurring in nearly one in ten women. Mammography is an important type of medical imaging used to screen healthy women for small curable breast cancers. X-ray images depend on differences in x-ray stopping power (attenuation) to separate tissues. In general, a clear separation between normal functioning tissue, and abnormal cancerous tissues is not possible since their attenuation is very similar. However both functional tissue and cancer can be separated from fatty storage tissues which normally surround active breast tissue, even in lean women. This is due to a substantially lower attenuation caused by fat. In older women, the functional glandular tissue diminishes, leaving only thin supporting tissues clearly outlined by fatty tissues. Mammography in these "mature" breasts is very effective, since even small cancers are well outlined by fat. In addition, many cancers develop calcium deposits which strongly stop X-rays and are easily seen on mammograms.

1.2 Analysis of Mammograms.

The image created by the X-ray process can reveal benign or malignant bodies or known predispositions to malignant bodies such as micro-calcifications. The shape of the bodies being the main visual factor to its potential malignancy. Round, oval and lobular shapes are indeterminate, however irregular more pointed bodies are of more

concern because it implies indistinct margins which are more often malignant (tumor infiltrating edges). Processes that scar the breast are often irregular. Micro-calcifications are also of concern as they are often high risk predispositions to malignant bodies. Often less visually obvious then the large tumours and difficult to detect. Due to this difficulty mammogram analysis is often carried out by way of comparison from the left breast image to the right. The images are placed on a lit view-box back to back creating the centre as a pseudo 'mirror' line.

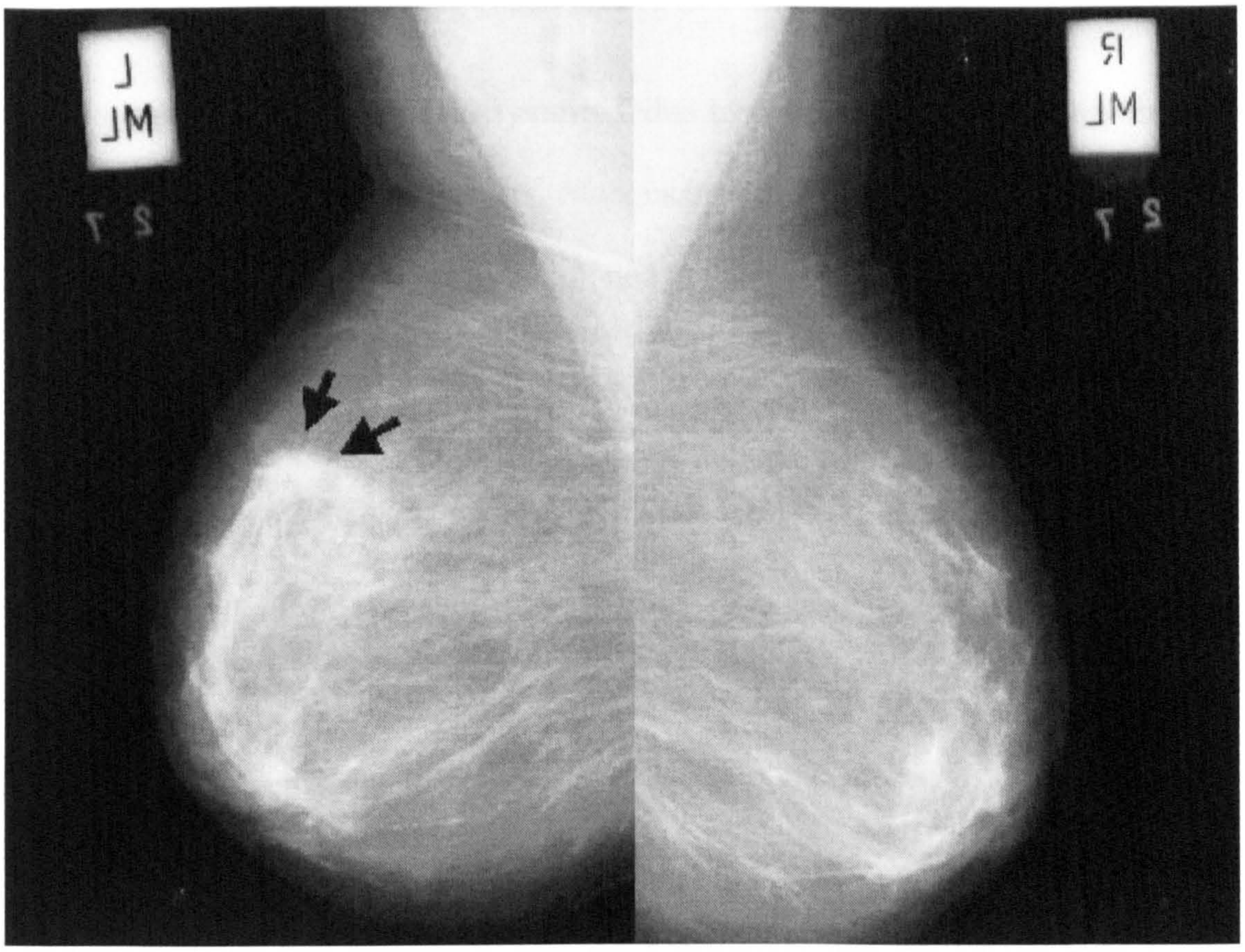
Figure 1.2.1 Mammogram Viewpoints



The two main viewpoints shown in the above figure are lateral-dorsal (left) and cranio-caudal (right). While exact mirror images are not to be expected, from an overall vantage point the tissue patterns within each breast should be similarly distributed. An asymmetric area may be indicative of a developing mass, a variation of normal breast tissue.

Figure 1.2 gives an example of a mass formation in the left breast detected this mirror viewing approach.

Fig 1.2.2 Typical Positioning of Mammograms During Clinical Screening



Although useful, this method is not standard and expected practice for clinicians but rather a suggested practice. However this implies the role of symmetry as a potential indicator to cancer development.

1.3 Symmetry and Biological Fitness

In Evolutionary Biology anatomical symmetry is known to be an indicator for fitness and developmental stability. Past studies [71][83] show asymmetry or fluctuating asymmetry (FA) as a factor effecting various areas in evolutionary biology. The ideal phenotype for areas where there are two of a part is deemed to be an exact mirrored replica. Especially in the case of external parts of the body. The unconscious way in

which humans find a more symmetrical person more attractive than a less symmetrical person gives rise to the idea that symmetry is an indicator of health that has been evolved.

Deviation from this ideal symmetry due to developmental instability may be one of the predispositions of cancers. Malignant bodies develop in a highly unstable manner causing irregular shapes as mentioned in the previous section.

1.4 Breast Cancer and Asymmetry.

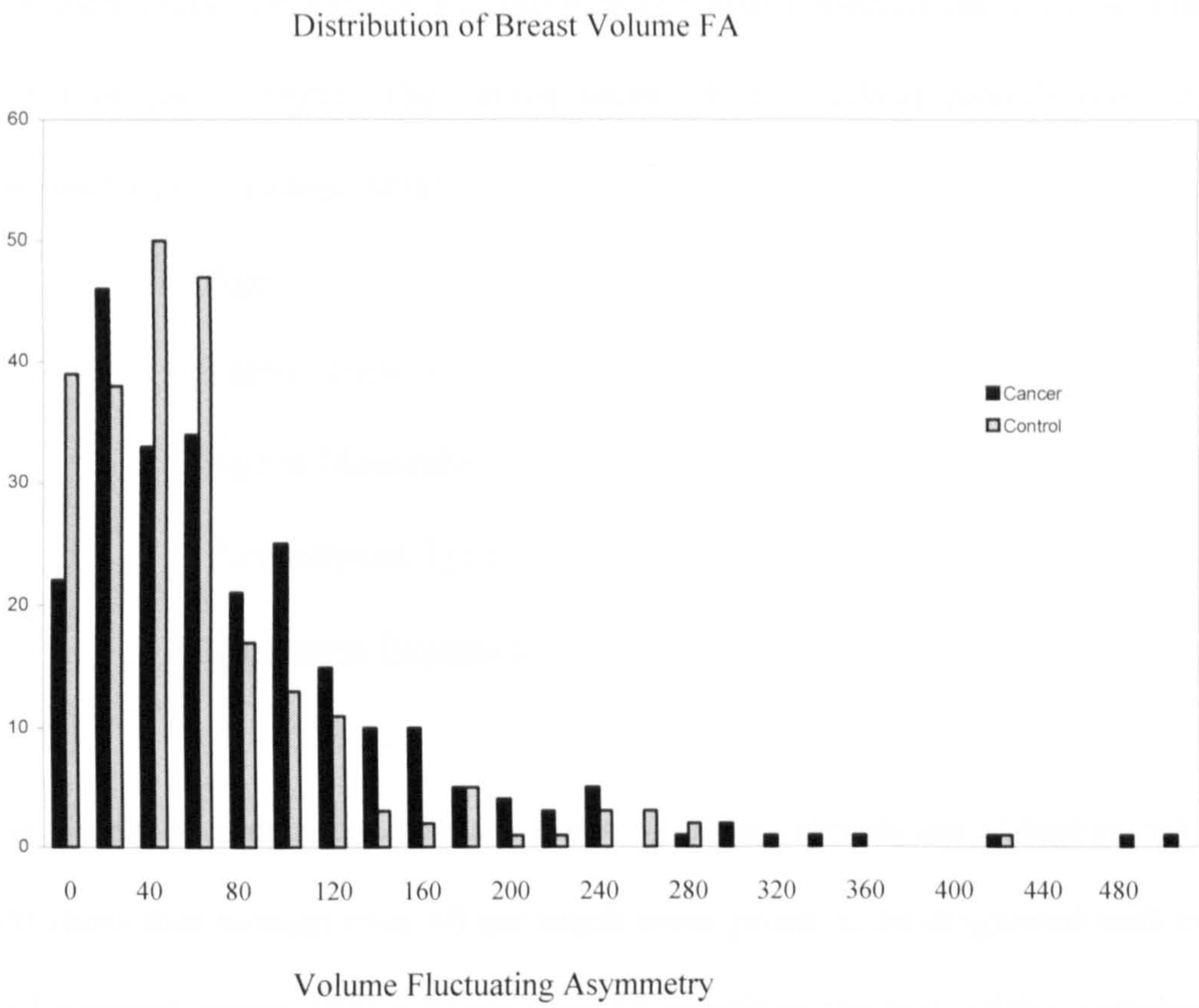
Consequently studies relating phenotypic asymmetry to conditions in health have been carried out [55][56]. *Scutt* [83] in 1998 gave statistical studies of measured asymmetry of various phenotypic traits and compared them to breast cancer occurrence. This study focuses on external traits of the breast, using cranio-caudal mammograms the largest width and maximum height measured. From this an estimated volume was calculated with the assumption that the curved edge of the cranio-caudal mammogram was parabolic.

The mammograms for *Scutt* [83] come from a major survey of women who had volunteered for a Liverpool study on breast cancer risk factors, the survey spanned a decade and took place from 1978 to 1988. This is also the same database used in this project. No diagnosed tumours are in any image at the time screening and all cases that are known to have developed cancer in later years serve as the cancer group and those that did not serve as the control group.

The fluctuating asymmetry (FA) in this case is the comparison of the two measured dimensions (cranio-caudal height and width) and the estimated volumes. *Scutt* showed that for breast cancer and the breast dimensions that there was no significant difference between the cancer developing subjects and the control subjects,

except for the most asymmetric cases. The occurrence of high asymmetric cases were shown to be higher in the cancer set.

Fig 1.4.1 Comparison of Estimated Breast Volume Difference [83]



The study [83] that shows the result in Fig 1.4.1 used 244 cases for both cancer and control sets. At the time mammography all cancer cases were diagnosed as disease free and subsequently went on to develop cancer at a later date. The volume FA is the estimated volume of the difference between the left and right breasts over the sum. FA here is regarded as relative and the size of the subject is taken into account in order to give this comparison.

1.5 Other Cancer Predispositions.

Due to the open system nature of human physiology cancer risk assessment is a difficult and complicated task. Breast cancer is no exception and is known to have a variety of both strong and weak predispositions. The survey mentioned above not only took mammograms of the subjects but also collected the medical history and details of each subject. The details taken from medical records most related to predisposition of cancer were:

Age

Family History

Age at Menarche

Parenchymal Type

Oestrogen Exposure

It is widely accepted that age is a factor as older subjects are at higher risk. Studies [55] show that women over 40 are much more prone to be diagnosed with cancer. In the Liverpool survey [83] subjects over 45 constitute the bulk of the participants. The family history relates to the subject's most closely related female family members and their known history of breast cancer development or non development. This is simply the number of people known to have been diagnosed with cancer. This is known to be an important as it is closely related to the hereditary genotype of the subject. Age at menarche is the age at which the subject's menstrual cycle began. Studies indicate this to be an important factor and is closely related to Oestrogen exposure which is a well known predisposition. The Parenchymal type is the internal breast tissues type known as the Parenchyma. This is divided into 4 categories, N1, P1, P2 and DY. Its is rare for a subject to have the left breast tissue having a different parenchyma to the

right. The four categories relate to the density and composition of the breast tissue, N1 being the most fatty. P1 and P2 to a lesser extent. DY describes Mammary Dysplasia, which are sheet like areas of high density.

1.6 Cancer Risk Assessment Using Artificial Neural Networks.

Since the completion of the Liverpool survey in 1988, to date all further diagnosis of cancer have been recorded using the North West Cancer Registry. This gives a database of prior details of cases and their outcomes and a suitable data set for non-linear classifiers such as Artificial Neural Networks (ANN).

Classifiers have been extensively used in relation to breast cancer. In Mammography the area of micro-calcification detection [105] is the most widespread area of ANN usage. This is due to micro-calcifications being difficult to see with the naked eye and are often overlooked by clinicians. Other investigations [41][99] include attempts to classify the shape of a detected tumour body as benign or malignant by inspecting the shape. Irregular shapes are more likely to be malignant.

This project uses more generalised data and looks at the risk assessment at a more comprehensive level. Although this is less specific and deterministic there are two main reasons for such an attempt. The first being the advantage of early recognition of a high risk case. The second being the reduction of false negatives in cancer screening programs that process large numbers of mammograms. Clinicians are faced with the difficulty of quickly processing each case due to large numbers, whilst being thorough in their inspection.

This project implements standard and non standard ANN architectures. The standard being the Multi Layer Perceptron (MLP). The non standard is the Conic Section Function Neural Network (CSF). This architecture is a novel hybrid of MLP

and Radial Basis Function (RBF) neural networks. CSF allows for a highly adaptable decision making process and is a more generalised form of ANN. A highly general and non-linear classification problem such as this may require a versatile ANN and learning process.

CHAPTER 2

Artificial Neural Network Theory

2.1 Information Processing Based on Biological Neurons.

The architectures of Artificial Neural Networks are inspired by the anatomical structure of neurons and connections within the human brain. The brain itself is a vastly complicated system consisting of neural cells or neurons and their interconnecting pathways known as synapses. The number of neurons and synapses are estimated to exceed 100 billion. Electrical pulses with varying degrees of intensity are fired from each neuron to others. The human brain averages 100 of these operations per second. In comparison Artificial Neural Networks (ANN) also consist of processing units called neurons and interconnecting pathways, however at a vastly simplified scale.

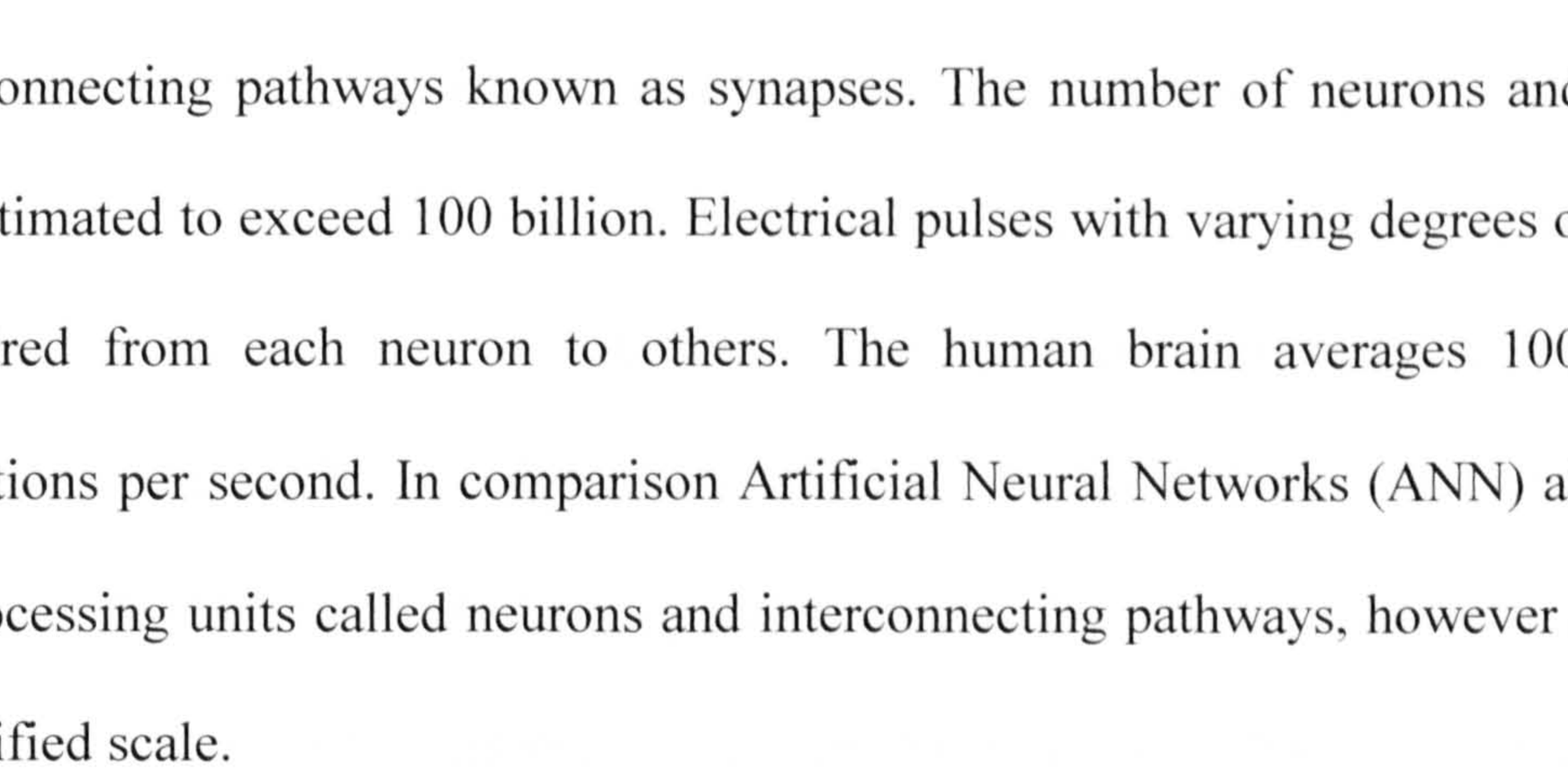


Figure 2.1.1 Biological Neuron

Direction of Pulses

Early work was done by McCulloch and Pitts [69] in 1943 using interconnected logic circuits, and Rosenblatt in 1957 [16] attempting to model the properties of biological neurons. However due to lack of computer facilities such models did not lead to successful applications but did give the foundation for ANN research later.

2.2 Feed Forward Neural Networks.

Depending on the type of ANN the neurons and pathways have different qualities associated with it and operates on the data that is processed accordingly. The subset of ANN used in this project are feed-forward neural networks [10]. The governing feature of feed-forward neural networks is that signals are processed without any feedback loops. The feed-forward ANN consists of a series of layers, each layer being a set of neurons. Data is first presented into the Input Layer, the information is processed by the input neurons according to a function quality within each neuron. Each input neuron is connected to every neuron in the next layer. The information is further processed by each connection or pathway, which each have their own quality. Pathway qualities are usually singular values known as weights of which the pathway signal is multiplied by.

2.2.1 Single Layer Feed Forward Networks.

The most basic form of feed forward neural network is the single layer network [10][16]. This form consists only of a set of input nodes with pathways leading to one set of neurons. The single layer term refers to this one set of neurons. The transfer

function within each neuron in this layer gives an output. This layer is known as an output layer of neurons. The input nodes are not counted as a layer because no computation takes place.

2.2.2 Multi Layer Feed Forward Networks.

Similar to Single Layer Feed Forward Networks, Multi Layer Feed Forward Networks [10][16] comprise of a set of input nodes and an output layer of neurons. However before the output layer are one or more hidden layers of neurons in the case for Multi Layer Networks. Figure 2.2.1 shows an example of a fully connected Multi Layer Feed Forward Network architecture with one hidden layer.

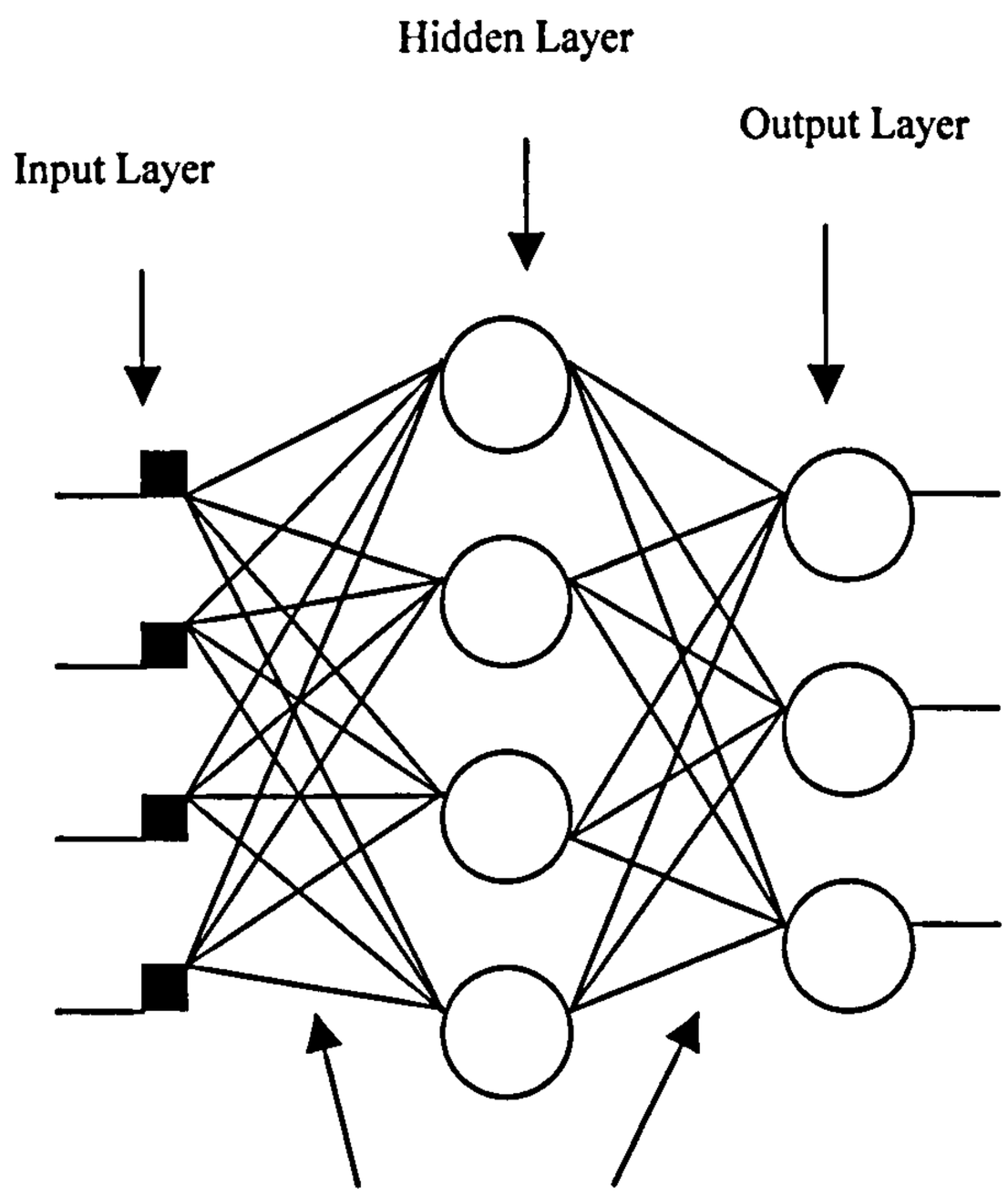


Figure 2.2.1 Fully Connected ANN

The term 'fully connected' describes the pathways between each layer [16]. All input nodes are connected to every hidden layer neuron and each neuron has a connection leading to every neuron in the next layer. If a problem has output patterns that are vastly different from it's input patterns or similarly if a problem is non linearly separable the addition of hidden layers can be advantageous.

The input value from each input node is multiplied by a weight w associated with each pathway. If the weight of the connection from input node i to neuron j was w_{ij} and the transfer function of neuron j was f_j then the process for neuron j is given as shown in Eq. (2.1).

$$y_{pj} = f_j(\sum_i w_{ij} x_{pi} + \theta_j) \tag{2.1}$$

where y_{pj} represents the output of neuron j for input pattern p. The θ_j term refers to a bias value associated with neuron j. The output y_{pj} is further propagated into the next layer where the same process takes place. y_{pj} is multiplied by the weight associated with the connection in the next layer and then added to the sum of the products of the weights and outputs of that layer. This sum is then the input for the transfer function of a neuron in the next layer.

2.2.3 Feed Forward Network Learning.

Training feed forward networks require algorithms that adjust the weight and bias terms so that the outputs of the network are correct for the given input pattern. There are numerous different types of algorithm that can achieve this. The two main categories of learning algorithm are Supervised and Unsupervised methods. The

supervised methods are based on training the network using a known set of desired outputs with corresponding input patterns. These desired outputs are compared to the actual outputs given by the trainee network. There are various methods to minimize the difference between the actual outputs and desired outputs. Details of such methods will be given in further sections. Unsupervised learning does not use desired responses and are based on self-organising methods [16]. These training processes adjust the weights and biases to give similar outputs for similar inputs.

2.2.4 The Delta Learning Rule.

This is an example of supervised learning for feed forward networks [10]. A network with a single output unit the outputs is

$$o = \sum_j w_j x_j + \theta \tag{2.2}$$

The function that gives the measure of the discrepancy between the networks outputs and the desired is based on the sum of the square of the differences, giving an error margin E, is

$$E = \sum_P E^P = \frac{1}{2}\sum_P (d^P - o^P)^2 \tag{2.3}$$

Where d is the desired output and o is the actual output for a set of input patterns indexed by P. The method known as gradient descent uses E^P to create a suitable adjustment to the weights proportional to the negative of the derivative of the error for

each input pattern with respect to each weight. This is shown in Eq. 2.4 where γ is a constant of proportionality.

$$\Delta_p w_j = -\gamma \frac{\partial E^P}{\partial w_j} \tag{2.4}$$

Eq. 2.4 is derived from

$$\frac{\partial E^P}{\partial w_j} = \frac{\partial E^P}{\partial o^P} \frac{\partial o^P}{\partial w_j} \tag{2.5}$$

From Eq. 2.3

$$\frac{\partial o^P}{\partial w_j} = x_j \tag{2.6}$$

and

$$\frac{\partial E^P}{\partial o_j} = -(d^P - o^P) \tag{2.7}$$

gives

$$\Delta_p w_j = \gamma (d^P - o^P) x_j \tag{2.8}$$

2.3 The Multi-layer Perceptron.

The Multi Layer Perceptron's (MLP) topology is that of a fully connected multi layer feed forward network. A layer of input nodes connected to one or more hidden layers of neurons in turn connected to the appropriate number of output neurons. MLP's are the most widely applied form of Multi Layer Feed Forward Network, especially for non-linear classification problems and pattern recognition.

The main characteristic of the MLP is the non-linear transfer function within the hidden neurons and the output neurons. This function is known as the sigmoidal activation function [16] defined by

$$y_j = \frac{1}{1+\exp(-\lambda x_j)} \tag{2.9}$$

Where y_j is the output for neuron j in either a hidden or the output layer and x_j is the input of that neuron. In MLP's as with all fully connected topologies this input is the sum of the values entering the neuron from all nodes in the previous layer. The λ term is a biasing constant that can be varied to alter the steepness of the sigmoidal activation function. In many cases this constant is set to 1.

2.3.1 Multi-Layer Perceptron Learning.

Supervised learning [10][16] for the MLP can be implemented by categorising the process in two parts. The first is known as the forward pass of the MLP. This is introduction of an input pattern to the input nodes and the inputs values thus processed by the set weights and biases to the hidden layer and then to the output layer. The output is then obtained. The second category for supervised learning is the backward pass. This is the adjustment of the weight and bias values determined by the training algorithm using the output obtained from the forward pass.

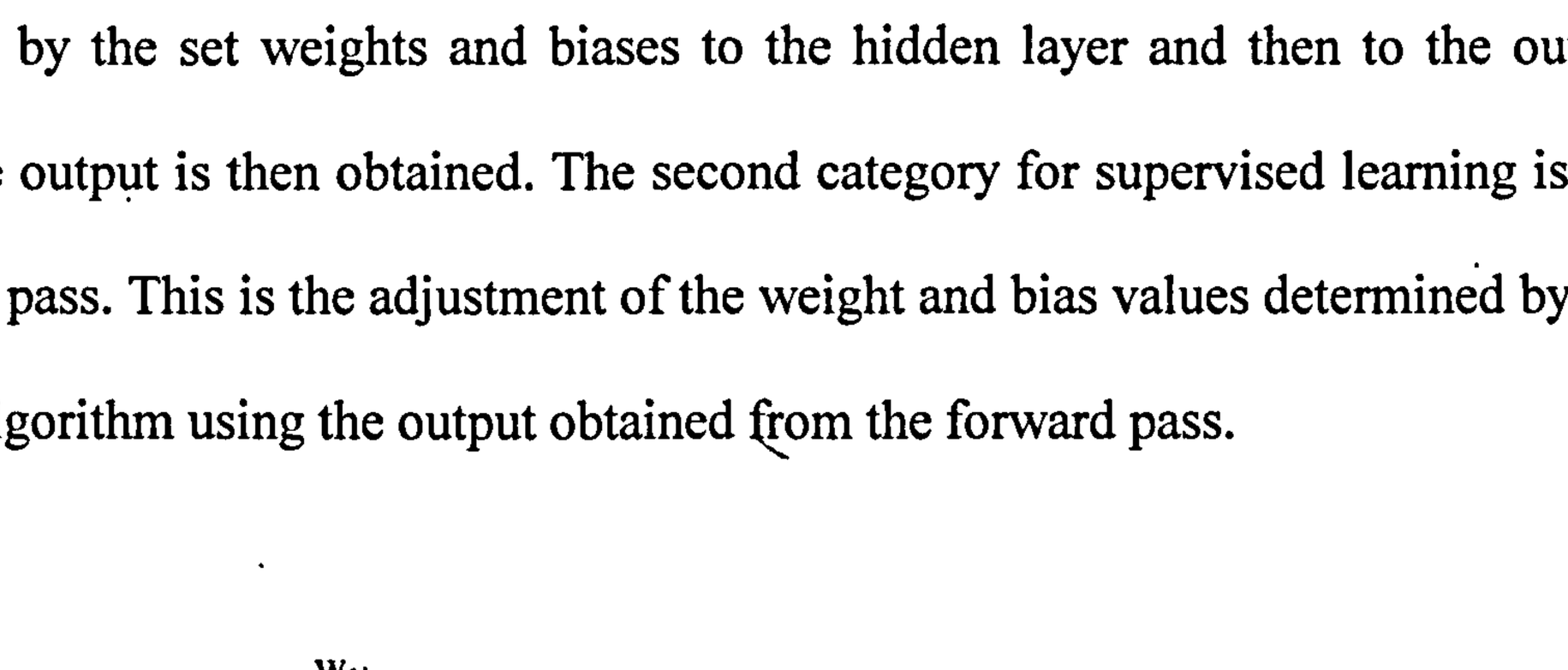


Figure. 2.3.1 ANN Neuron

Figure 2.3.1 shows in detail the composition of one neuron j with n number of inputs.

The forward pass process for this neuron j is

$$o_j = (\sum_i^n w_{ij} + y_i) + t_j \tag{2.10}$$

The evaluation of y_j is calculated using the sigmoidal activation function given in Eq. 2.9. Once a forward pass is complete and outputs obtained the required weight adjustments for the output layer can be calculated using the supervised learning algorithm which will be given detail in further sections. The algorithm adjusts the output layer weights first then the hidden layer weights. If there is more than one hidden layer the last hidden layer is adjusted first and then the layer preceding it. Thus the term backward pass. The complete procedure is summarised in the following figure.

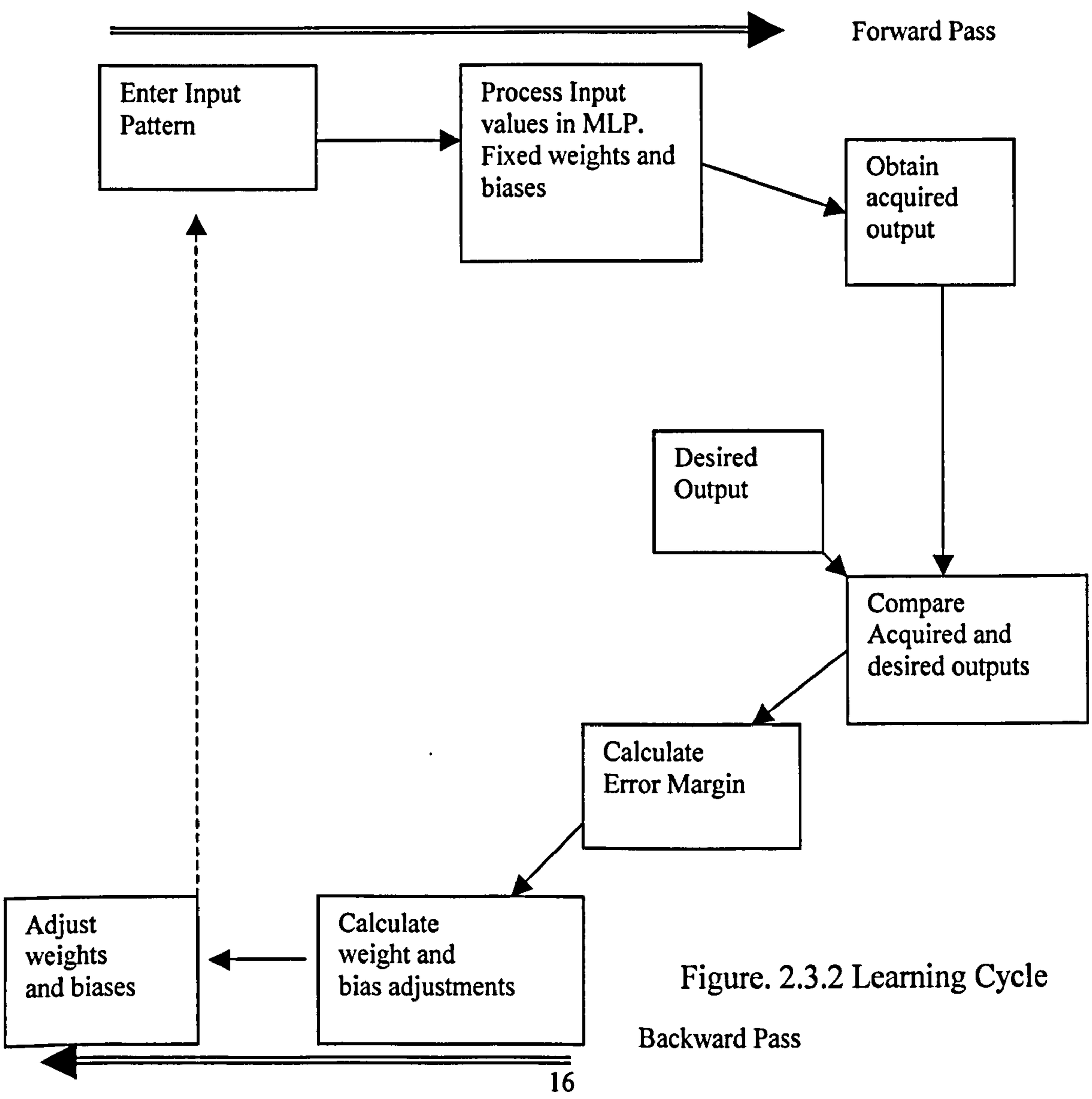


Figure. 2.3.2 Learning Cycle

2.3.2 The Error Back Propagation Training Algorithm

This commonly used supervised learning algorithm calculates the adjustments required in the backward pass of the MLP training cycle. The Error Back Propagation Algorithm (EBP) [10][16] is an iterative process that seeks to minimise the error margin between the acquired outputs of the MLP and the desired. The EBP repeats until the error margin is reduced to a satisfactory level. First the MLP is initialised with a set of random weights and biases. Then as in figure 2.3.2 the first set of input patterns are entered and the outputs are obtained processed by the random weights and biases. The error is then calculated and used to evaluate the necessary adjustments to each weight.

2.3.3 The Generalised Delta Leaning Rule.

This is the computation method for evaluating the weight and bias adjustments for the backward pass procedure in EBP, known as the Generalised Delta Learning Rule [10]. Since MLP use non linear the sigmoidal activation functions which is a differentiable function of the input

$$a_i^p = f(i_i^p) \tag{2.10}$$

Where

$$i_i^P = \sum_j w_{ij} a_j^P + \theta_i \tag{2.11}$$

The error margin E is calculated from each of the differences between the actual outputs at the output neurons and the respective desired output.

$$E = \sum_p E^p = \frac{1}{2} \sum_p \sum_i (d_i^p - a_i^p)^2 \tag{2.12}$$

$$E^p = \frac{1}{2} \sum_i (d_i - a_i)^2 \tag{2.13}$$

The desired outputs are d_i and E^p is the error margin for pattern p. The derivative of the error with respect to the weight is.

$$\frac{\partial E^p}{\partial w_{ij}} = \frac{\partial E^p}{\partial i^p} \frac{\partial i^p}{\partial w_{ij}} \tag{2.14}$$

Equation 2.11 can be arranged to match the second factor of equation 2.14 as

$$\frac{\partial i_i^p}{\partial w_{ij}} = a_j^p \tag{2.15}$$

The error signal derivative is

$$\delta_i^p = -\frac{\partial E^p}{\partial i_i^p} = -\frac{\partial E^p}{\partial a_i^p} \frac{\partial a_i^p}{\partial i_i^p} \tag{2.16}$$

The first product in Equation 2.16 can be written as

$$\frac{\partial E^p}{\partial a_i^p} = -(d_i^p - a_i^p) \tag{2.17}$$

The second product in Equation 2.16 is the derivative of the activation function f for neuron i.

$$\frac{\partial a_i^p}{\partial i_i^p} = f'(i_i^p) \tag{2.18}$$

Substituting Eqs. 2.17 and 2.18 into 2.16, the error term can be written as

$$\delta_i^p = (d_i^p - a_i^p) f'(i_i^p) \tag{2.19}$$

This leads to an adjustment value

$$\Delta_P w_{ij} = \gamma \delta_i^p a_i^p \tag{2.20}$$

It is proportional to the product of the error term on neuron *i* and the output of neuron *j* connected to it. The sigmoid activation function in Equation 2.9 with λ set to 1, can be derived to

$$f'(i_i^p) = a_i^p(1-a_i^p) \tag{2.21}$$

Substituting this into Equation 2.19 the error term can be defined as

$$\delta_i^p = (d_i^p - a_i^p)a_i^p(1-a_i^p) \tag{2.22}$$

For the hidden layer neurons the error term is calculated using the error terms of the neurons that it is connected to and the weights of those connections.

$$\delta_i^p = a_i^p(1-a_i^p)\sum_h \delta_h^p w_{hi} \tag{2.23}$$

2.3.4 Adaptive Learning Rate and Momentum

In equation 2.20 the γ term known as the learning rate can also be adapted during training. Without an adaptive γ a constant is chosen, a small γ can cause the training to be slow however a γ that is too large can cause instability and oscillation. Using an adaptive learning rate that is dependant on the past weight change using a term known as momentum α [6].

$$\Delta\ w_{ij}(t+1) = \gamma\delta_i^p a_i^p + \alpha\Delta w_{ij}(t) \tag{2.24}$$

Where *t* is the number of the learning cycle.

2.3.5 Log Likelihood Error Function

Alternative to the error sum of squares error function defined in equation 2.13 the Log Likelihood Error function [10] is more appropriate for classification problems with binary network targets. For a two class problem with classes with C_1 and C_2 a single output a and input vector $\boldsymbol{x}$. If value y is considered to be the posterior probability for C_1 this can be expressed as

$$a = P(C_1|\boldsymbol{x}) \tag{2.25}$$

Consequently the posterior probability for the second class can be expressed as

$$1\text{-}a = P(C_2|\boldsymbol{x}) \tag{2.26}$$

If the desired output d for class C_1 is 1 and C_2 is 0 the posterior probability expression can be combined so that the probability for obtaining either desired target is

$$p(d|\boldsymbol{x}) = a^d (1\text{-}a)^{1\text{-}d} \tag{2.27}$$

The likelihood of observing the training data set of size n can be expressed as

$$\prod_n (a^n)^{d^n} (1-a^n)^{1-d^n} \tag{2.28}$$

The error function is produced using this expression however it is computationally more sensible to use the negative logarithm of this expression as the error function giving

$$E = -\sum_{n} \left\{ d^n \ln a^n + (1 - d^n) \ln(1 - a^n) \right\} \tag{2.29}$$

2.3.6 Regularisation using Weight Decay

The problem of over fitting is one that frequently occurs when minimising the error function E in learning algorithms. The method of regularisation [10] uses a penalty term Ω that is added to the error function value. This term has the effect to stiffening the boundaries within the data space so that the learning algorithm does not so readily fit the boundaries specifically to the training data and some generality is preserved. The influence of the penalty term can be determined by parameter ν.

$$\tilde{E} = E + \nu\Omega \tag{2.30}$$

Evaluation of the penalty term in its most simple form is a method known as Weight Decay. The sum of the squares of all weights and biases in the network determines Ω in the weight decay method.

2.3.7 *N* - Cross Validation and the Receiver Operator Curve.

N Cross Validation [86][98][60] is widely used method for evaluating the performance of the learning algorithm. The data set used in training can be split into an *N* number of equal blocks. Each block in turn serves as a validation set and the remaining serve as the training set. Each block must contain patterns belonging to both cancer and non cancer sets.

Total number of both true and falsely identified outputs from the *N* validation sets are evaluated. Using a variable acceptance threshold within the output range the varying

number of false positives and false negatives are obtained. The size of which depends on the number of acceptance thresholds used. From this two variables are derived, first is Sensitivity defined as:

$$Sensitivity = \frac{TruePositives}{TruePositives + FalseNegatvies} \tag{2.31}$$

and Specitivity:

$$Specitivity = \frac{TrueNegatvies}{TrueNegatives + FalsePositives} \tag{2.32}$$

If plotted as Sensitivity (vertical axis) against 1- Specitivity (horizontal axis) using the variable number of thresholds a characteristic called the Receiver Operator Curve is formed [98]. This curve demonstrates the ability to discriminate. An ideal classifier would produce a curve tending close a right angle at the top left corner of the axis. A random guess with an equal number of true and false outputs would produce a straight diagonal line from the origin to the top right corner of the axis.

CHAPTER 3

Conic Section Function Neural Network

3.1 Introduction to Conic Section Function Neural Networks

Dorffner [25] describes a hybrid neural network that uses the properties of Multi Layer Perceptrons and Radial Basis Function (RBF) networks. MLP and RBF networks are both feed forward neural networks that consist of fully connected input, output and hidden layers. The described hybrid neural network is called a Conic Section Function network (CSF). One of the major differences between MLP and RBF is the type of decision boundary that is placed in the input space. Conic Section Function (CSF) networks adopt both the MLP unbounded planar boundary *hyper-planes* and the RBF bounded radial boundary *hyper-spheres*. With the addition of an extra parameter that identifies the degree MLP and RBF characteristics, the CSF boundary has the ability to change from RBF form to MLP form or vice-versa. This ability to change boundary type gives the CSF network enhanced generality. Another advantage of having variable decision boundaries is that fewer boundaries are needed in the input space. Reducing the number of boundaries also reduces the number of hidden layer nodes and thus the network size.

3.2 Radial Basis Function (RBF) networks

Radial Basis Function Networks [67] differ from MLP networks principally in the transfer function of the neurons. Using non-linear radially symmetric functions, closed decision boundaries are created in the input space known as *hyper-spheres.* Each neuron is associated with one hyper-spherical decision boundary in the input space [25].

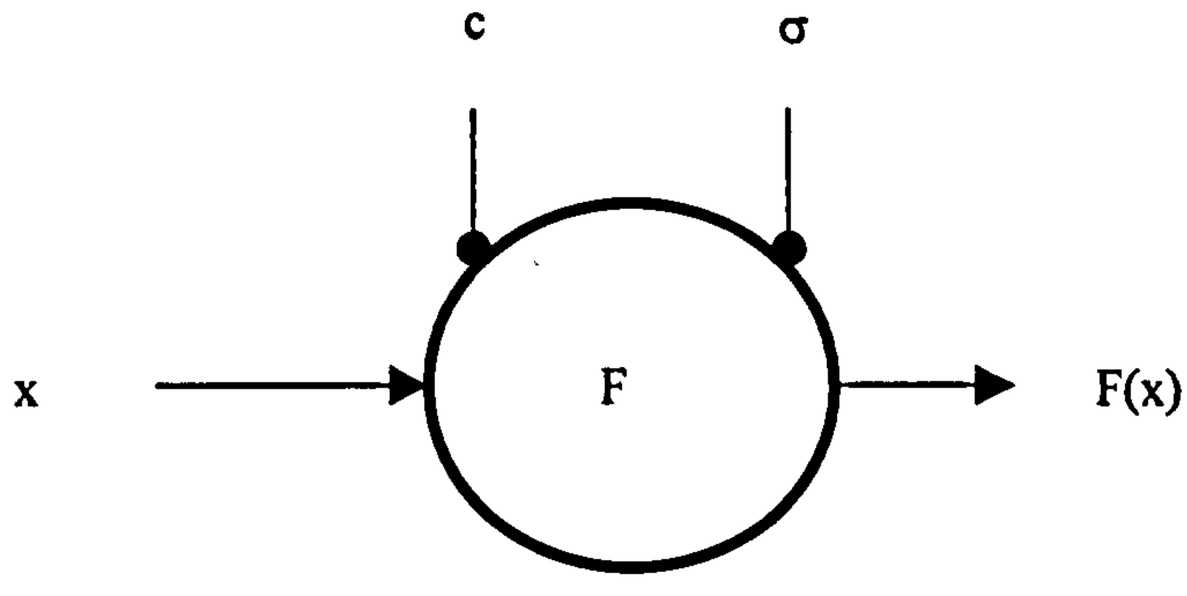


Figure. 3.2.1 RBF network neuron

Figure 3.2.1 shows a single RBF neuron with two associated parameters c and σ. When the input $x = c$ the function F(x) is 1 when the function F(x) processed by the neuron is defined as:

$$F = \exp[-\frac{(x-c)^2}{2\sigma^2}] \tag{3.1}$$

The c term determines the maximum value of the transfer function. As x deviates from c the output drops and becomes negligible as the x values become further from c. This function gives significant outputs only when x is over a certain range. Due to this RBF neurons are said to have a *receptive field.* This is determined by the σ term.

3.2.1 Radial Basis Function Topology

An RBF [67] network is also categorised as fully connected feed forward neural networks, consisting of one hidden layer, one input and one output layer. The hidden layer consists of neurons as shown in figure 3.2.1. The output layer is a linear summation of the outputs from all hidden layer neurons.

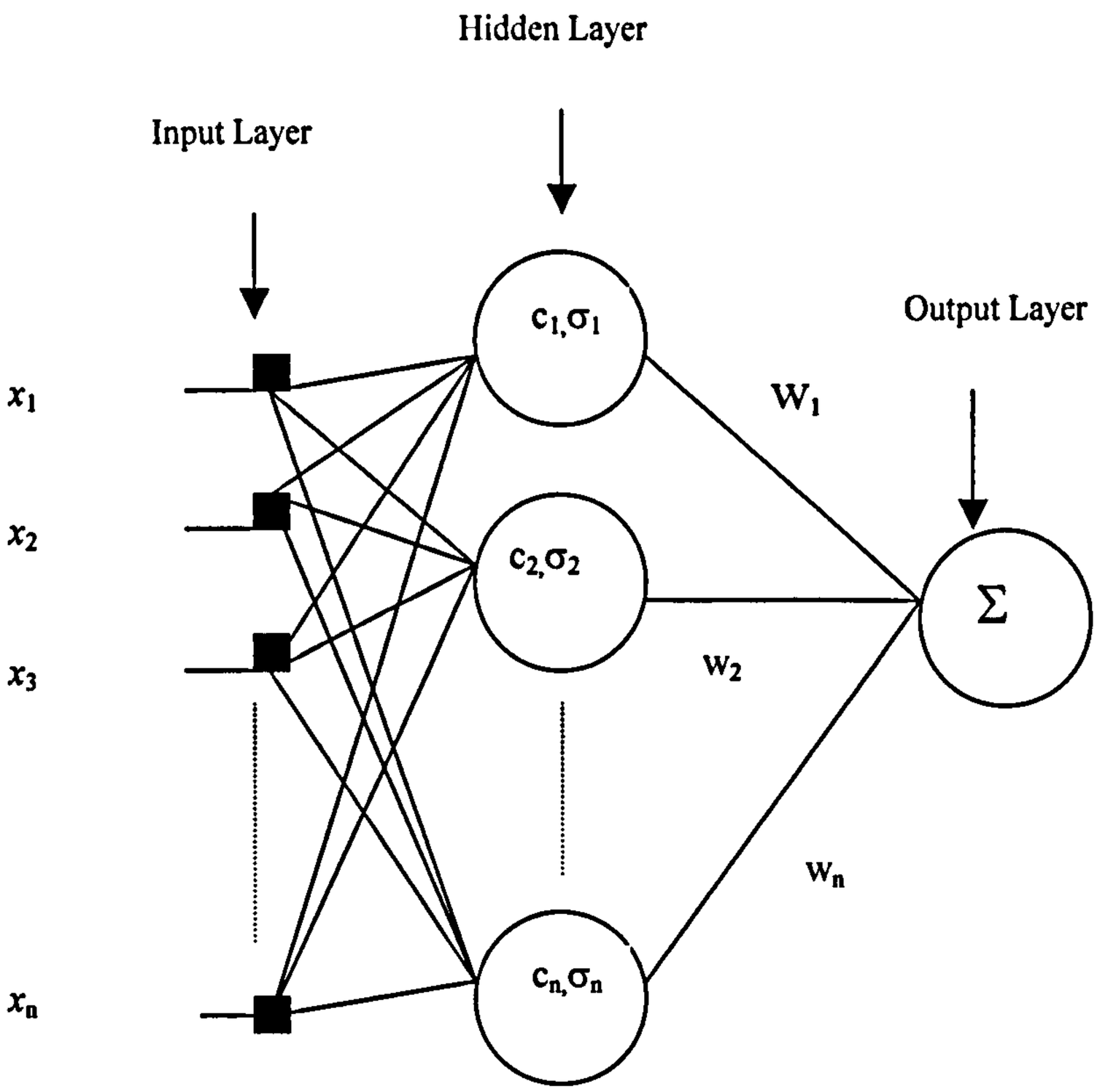


Figure. 3.2.2 RBF network architecture.

Unlike MLP networks the connections from the input layer to the hidden layer are not assocciated with weights. Computation of the input vector begins at the hidden layer

neurons. The distance between the input vector x and the maximum of the transfer function c is given as r.

$$r = \|x - c\| \tag{3.2}$$

Once this distance is computed a set of non linear functions known as *basis functions* are applied. The *basis functions* evaluate how close the inputs are to the centres of the receptive fields. The connections from the hidden layer to the output layer do have assocciated weights similar to MLP and evaluates a linear summation. The general form for an RBF network is

$$F(x) = \sum_{i=1}^{N} w_i \phi(\|x - c_i\|) \tag{3.3}$$

The ϕ term represents the basis function and w being the weight term for neuron i for N neurons. The c term again represents the centre of the basis function.

There are a variety of basis functions [21] that can be chosen for ϕ, typical choices are:

Piecewise linear	$\phi(r) = r$	(3.4)
Cubic	$\phi(r) = r^3$	(3.5)
Gaussian	$\phi(r) = \exp(-\frac{r^2}{\sigma^2})$	(3.6)
Thin Plate Splines	$\phi(r) = r^2 \log(r)$	(3.7)
Multi Quadratic	$\phi(r) = \sqrt{(r^2 + \sigma^2)}$	(3.8)

Where σ is the width or scaling parameter. The most widley used basis fucntion is the Gaussian function.

3.3 Conic Sections

Conic sections are geometric shapes produced by the intersection of a plane and a double circular cone. These intersections can be circles, ellipses, parabolas, or hyperbolas. For a circle the plane cuts completely across one of the cones and is perpendicular to the axis of the cone.

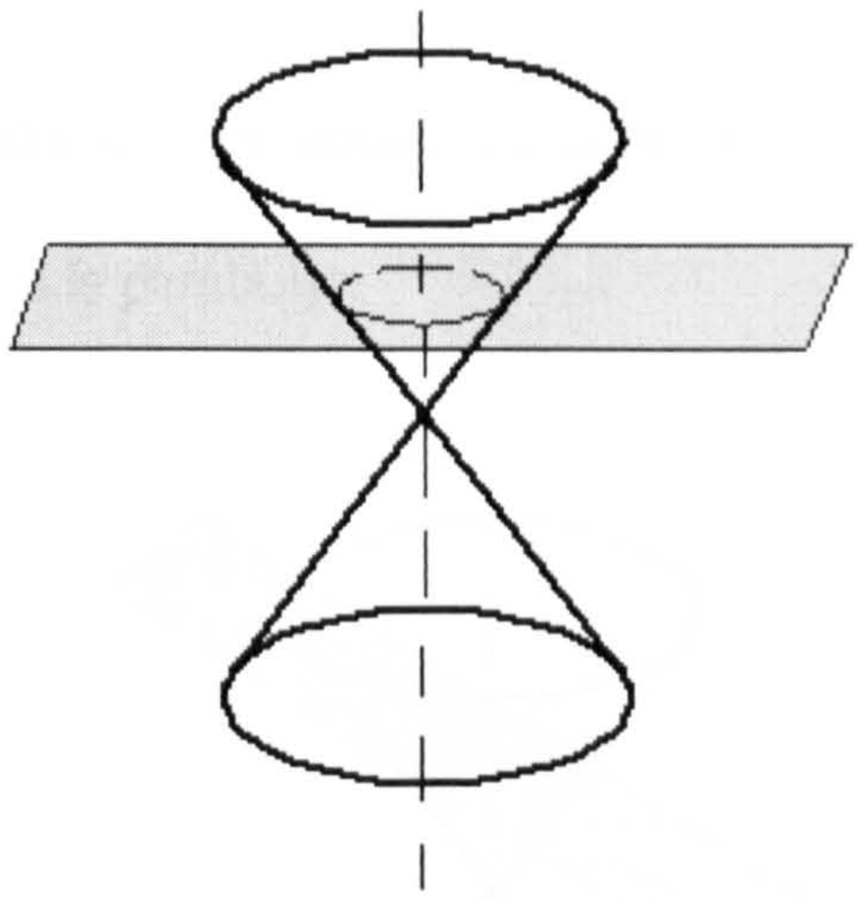


Figure. 3.3.1 Circular Conic Section

When then plane is not perpendicular to the axis the shape created on the plane is elliptical.

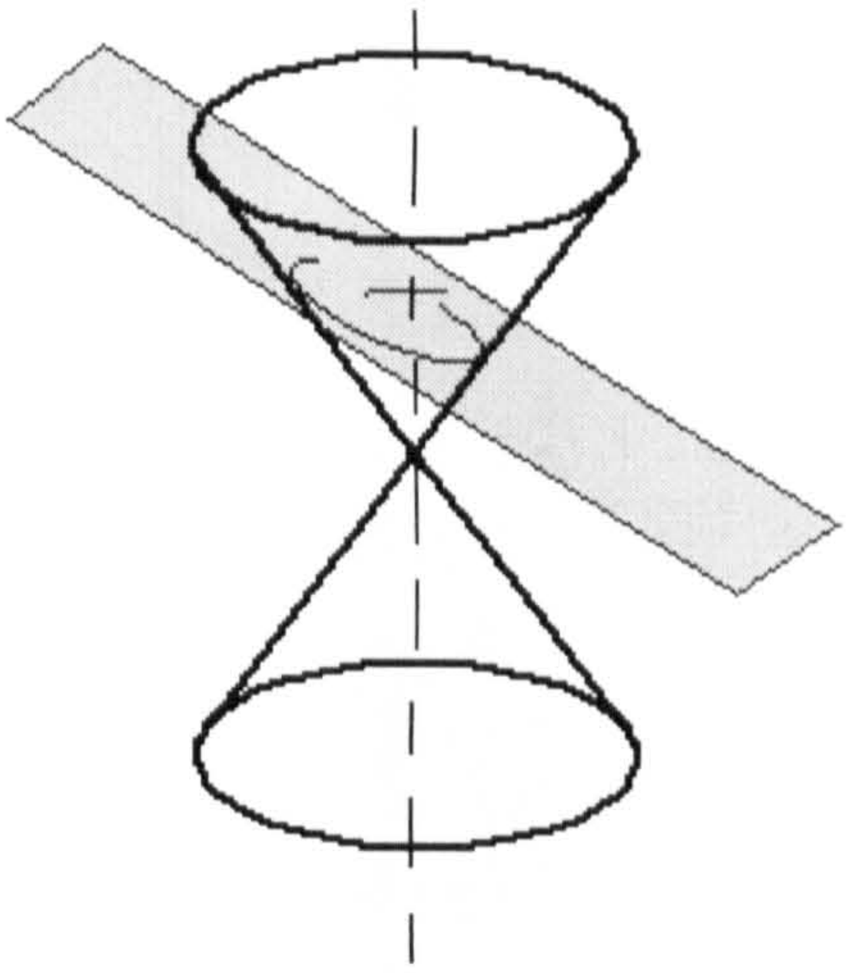


Figure. 3.3.2 Elliptical Conic Section

However If the plane doesn't cut across one entire surface or intersect both cones, the curve of the intersection is parabolic.

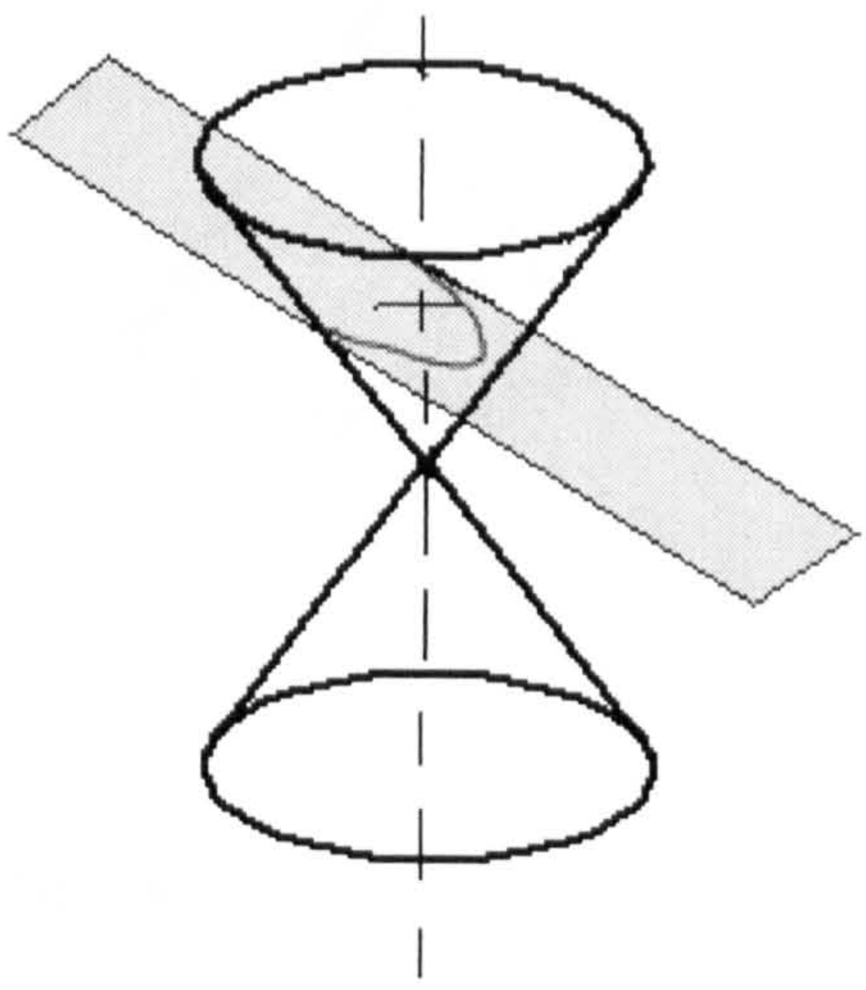


Figure. 3.3.3 Parabolic Conic Section

When the plane cuts through both of the cones parallel to the axis, the curve is hyperbolic.

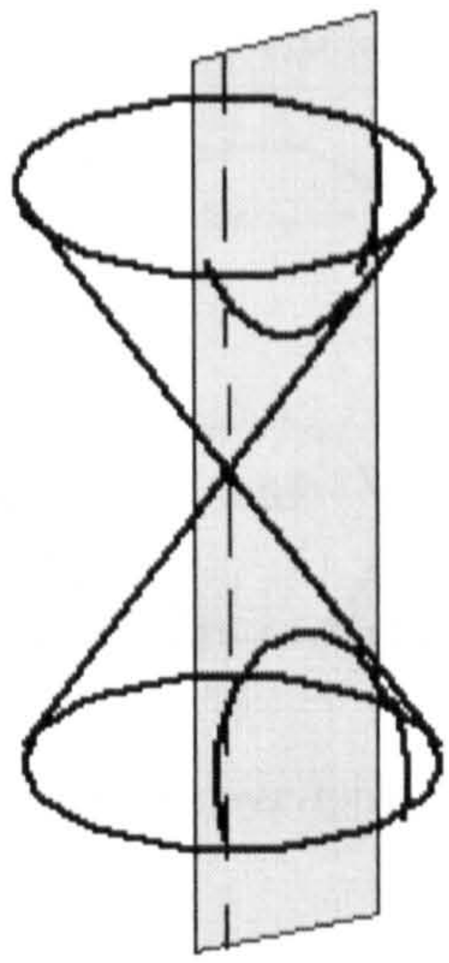


Figure. 3.3.4 Hyperbolic Conic Section

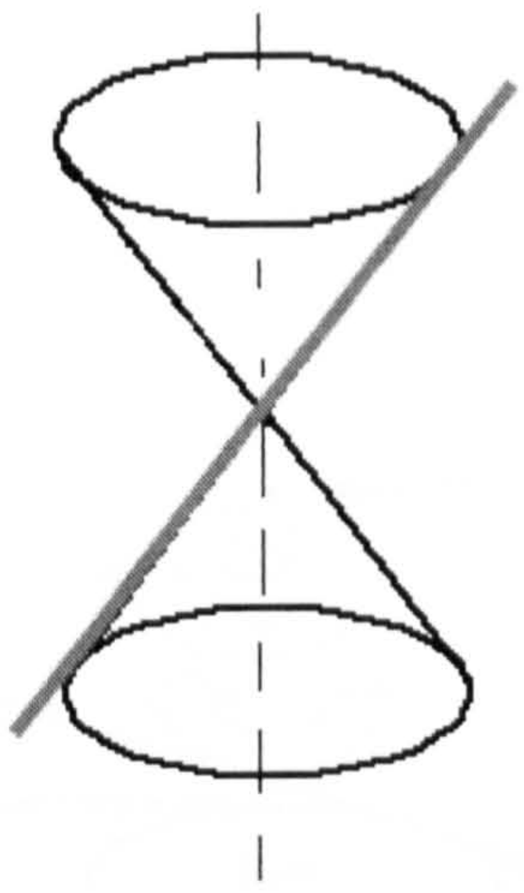


Figure. 3.3.5 Plane Conic Section

Using conic sections Dorfner [25] applies the variable shapes to form decision boundaries in the input space. Another method to define the conic sections is to introduce a variable known as *opening angle* 2ω. For example in the case the plane cutting the cone to form a circle as in figure 3.3.1 and the distance from the plane to the vertex of the cone is equal to that of the radius of the circle the half opening angle ω is 45 degrees.

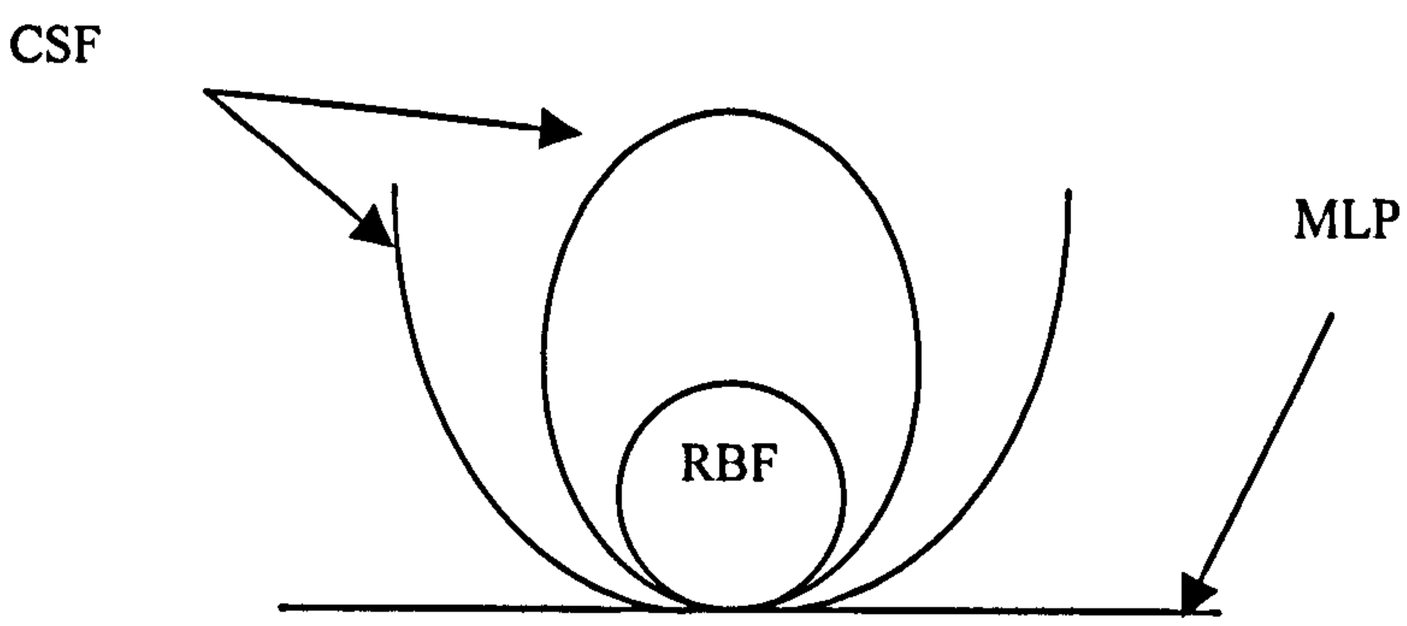


Figure 3.3.6 Conic sections

The shape of the conic section changes as ω varies. As the angle increases the circle (RBF characteristic) becomes and ellipse which in turn can become un bounded and become a parabola and then a straight line (MLP characteristic).

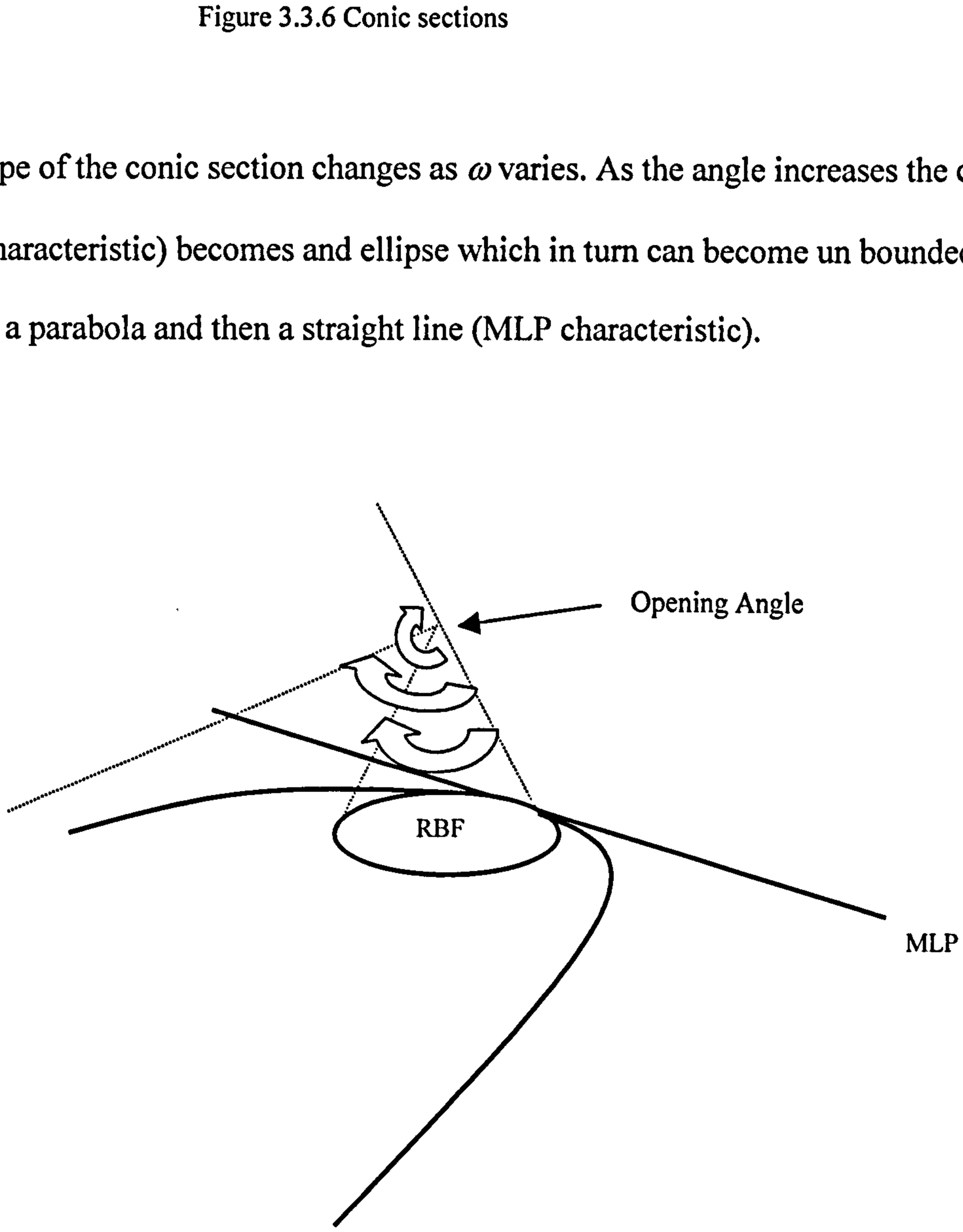


Figure. 3.3.7 Conic Section projections.

3.4 Conic Section Function Neural Network

The propagation rule for the feed forward process in CSF networks are a combination of MLP abd RBF propagation rules. Use of weights on all connections (MLP) and centre values in hidden to output layer connections (RBF). Dorfner [25] describes the charracteristics of CSF networks from the following derivation.

If x is a point on the surface of the cone, ω can be any value from $-\pi/2$ to $\pi/2$. If v is the vertes of the cone and a is a unit vector of the cone's axis then a circular cone can be desribed as:

$$(\vec{x}-\vec{v})\vec{a} = \cos\omega\|\vec{x}-\vec{v}\| \tag{3.9}$$

If the vectors and are written in co-ordinate pairs for two dimensional space the cone can be desribed as

$$(x_1-v_1)a_1+(x_2-v_2)a_2 = \cos\omega\sqrt{(x_1-v_1)^2+(x_1-v_2)^2} \tag{3.10}$$

For n dimensional space

$$\sum_{i=1}^{n+1}(x_i-v_i)a_i = \cos\omega\sqrt{\sum_{i=1}^{n+1}(x_i-v_i)^2} \tag{3.11}$$

This form gives the equation of the cone and input space intersection, if the co-ordinate system is set so that the n dimensions are that of the input space by setting $x_{n+1} = 0$. The co-ordinates of the centre terms c (RBF) can replace the co-ordinates of the vertex v. The distance between x and v is the radius of a circular conic section so

long as the opening angle 2ω is 90 degrees. Using this subsitution the equation can be written as

$$y_j = \sum_{i=1}^{n+1}(x_i - c_{ij})a_{ij} - \cos\omega_j \sqrt{\sum_{i=1}^{n+1}(x_i - c_{ij})^2} \qquad (3.12)$$

Which defines the propagation rule of the CSF network. The a_{ij} term represents the weight of the connections between the input and hidden layer (MLP). The c_{ij} term represents the centre co-ordinates (RBF) from input i to hidden neuron j. The y_j term represents the activation function for hidden neuron j.

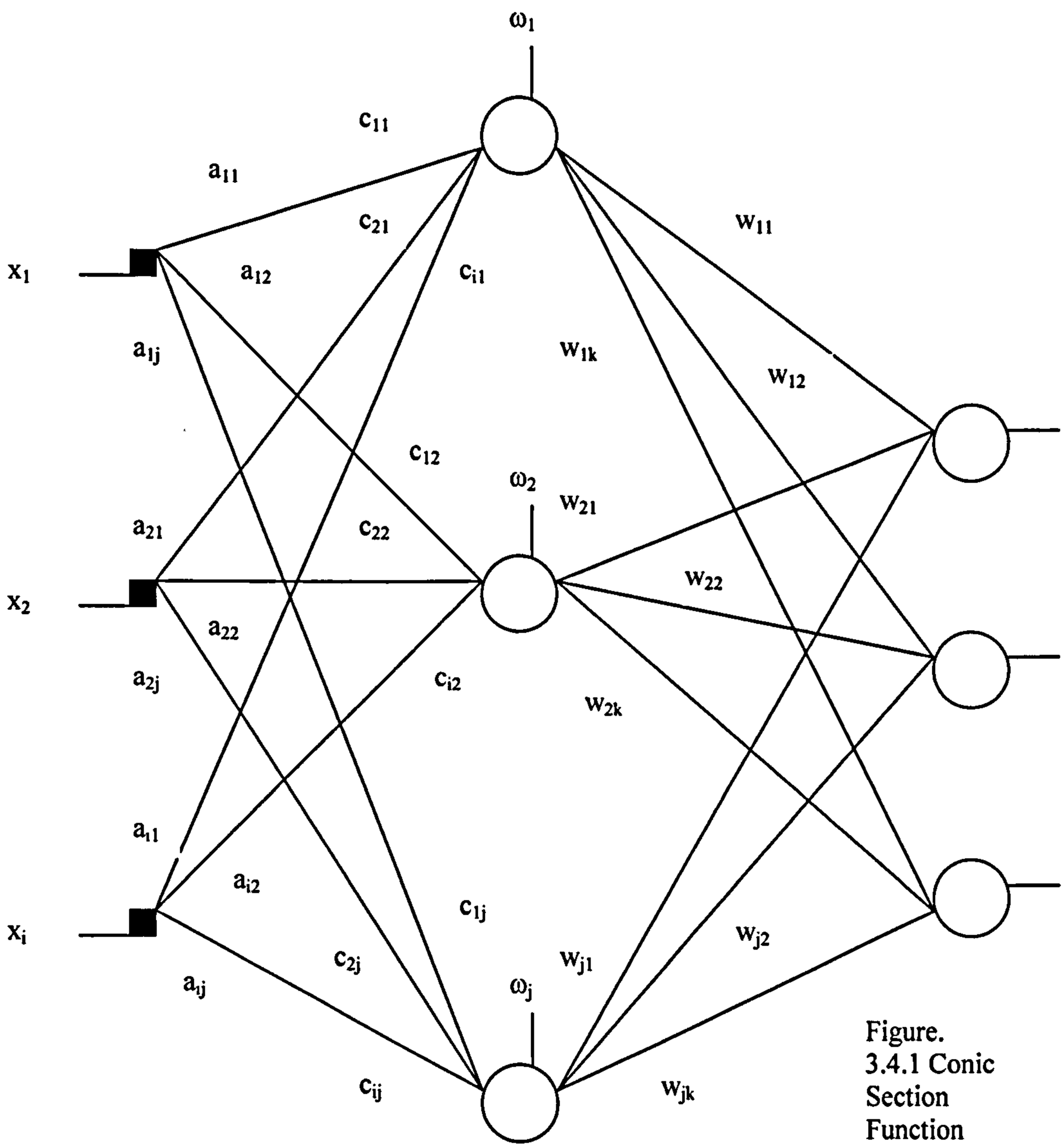


Figure. 3.4.1 Conic Section Function Network

3.5 Conic Section Function Network Learning

The learning process for CSF network training involves the determination of the hyper-sphere centre location and then updating the centres along with weights and opening angles. Within the data space the learning process can be considered to place a number of hyper-sphere (RBF) boundaries and then adjust their location and shape. This algorithm is a mixture of a RBF leaning algorithm and a form of EBP for the updating of centres weights and opening angles.

3.5.1 Radial Basis Function Leaning

The part of the CSF learning algorithm that deals with hyper-sphere and centre placement into the data space is known as the *Orthogonal Least Squares Learning Algorithm* (OLS) [21]. This method is method uses a block data set to select to determine the number hyper-spheres and their centre values. The RBF network is initialised with no hidden neurons. Each iteration of the OLS algorithm adds a hidden neuron until a satisfactory error margin is reached [21].

The OLS algorithm is derived from linear regression models, according to which a desired output *d(n)* is defined as:

$$d(n) = \sum_{i=1}^{M} p_i(n)\theta_i + e(n), \qquad n = 1,2.....N \tag{3.13}$$

where the θ_i are the model parameters, *e(n)* is the error signal and $p_i(n)$ are the regressors which are a function of *x(n)*:

$$p_i(n) = p_i(x(n)) \tag{3.14}$$

Converting equation 3.13 into matrix notation:

$$d = P\theta + E \tag{3.15}$$

where

$$d = [d(1), d(2), ..., d(N)]^T \tag{3.16}$$

$$P = [p_1, p_2, .., p_M] \tag{3.17}$$

$$\Theta = [\theta_1, \theta_2, .., \theta_M]^T \qquad 1 \le i \le M \tag{3.18}$$

$$p_i = [p_i(1), p_i(2), ..., p_i(N)]^T \tag{3.19}$$

$$E = [e(1), e(2), ..., e(N)]^T \tag{3.20}$$

The regressor vectors p_i are transformed into a corresponding set of orthogonal basis vectors so that contribution to the output energy from each basis vector w_i can be calculated, where w_i=[w_1, w_2..... w_M]. The regression matrix P can be defined as [21]:

$$P=WA \tag{3.21}$$

Where A is an M x M triangular matrix with 1's on the diagonal and 0's below the diagonal and W is an N x M matrix with orthogonal columns w_i where:

$$W^TW=H \tag{3.22}$$

Where H is the diagonal with elements h_i:

$$h_i = w_i^T w = \sum_{t=1}^{N} w_i(t) w_i(t), \qquad 1 \le i \le M \tag{3.23}$$

The orthogonal basis vectors w_i spans the same space as p_i and equation 3.15 can be defined as:

$$d=Wg+E \tag{3.24}$$

The orthogonal least square solution [21] is given by

$$g=H^{1}W^{T}d \tag{3.25}$$

where

$$g_i = \frac{w_i^T d}{(w_i^T w_i)} \qquad 1 \le i \le M \tag{3.26}$$

A classical method known as Gram-Schmidt computes one column of A at a time and then orthogonalises P in iterative steps counted by k. At the k^{th} step the k^{th} column is made orthogonal to each of the previously orthogonalised columns and the procedure is repeated until $k=M$.

The calculations are

$$w_1=p_1 \tag{3.27}$$

$$\alpha_{ik} = \frac{w_i^T p_k}{(w_i^T w_i)} \qquad 1 \le i < k \tag{3.28}$$

$$w_k = p_k - \sum_{i=1}^{k-1} \alpha_{ik} w_i \tag{3.29}$$

where $k=2.....M$.

In RBF training the data block $x(n)$ can be very large and only a subset of is required for adequate modelling. A smaller set of regressors M_s can be selected using the OLS algorithm implemented in a forward regression manner. If the sum of the squares of $d(n)$ is defined as:

$$d^T d = \sum_{i=1}^{k-1} g_i^2 w_i^T w_i + E^T E \tag{3.30}$$

and an error term due to w_i defined as

$$e_i = \frac{g_i^2 w_i^T w_i}{(d^T d)} \qquad 1 \le i \le M \tag{3.31}$$

then the forward regression procedure is as follows

The first step for $1 \le i \le M$

$$w_1^{(i)} = p_1 \tag{3.31}$$

$$g_i^{(i)} = \frac{(w_i^{(i)})^T d}{(w_i^{(i)})^T w_i^{(i)}} \qquad 1 \le i < k \tag{3.32}$$

$$[err]_i^{(i)} = \frac{(g_i^{(i)})^2 (w_{i_{(i)}})^T w_i^{(i)}}{d^T d} \tag{3.33}$$

then find

$$[err]_i^{(i1)} = \max\{[err]_i^{(i)}, 1 \le i \le M\} \tag{3.34}$$

and select

$$w_i = w_i^{(i1)} = p_{i1} \tag{3.35}$$

At the k^{th} step where $k \ge 2$, for $1 \le i \le M$, $i \ne i_i$ and $i \ne i_{k-1}$, compute

$$\alpha_{jk}^{(i)} = \frac{w_j^T p_i}{(w_j^T w_j)} \qquad 1 \le j \le k \tag{3.36}$$

$$w_k^{(i)} = p_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} w_j \tag{3.37}$$

$$g_k^{(i)} = \frac{(w_k^{(i)})^T d}{(w_k^{(i)})^T w_k^{(i)}} \tag{3.38}$$

$$[err]_i^{(i)} = \frac{(g_i^{(i)})^2 (w_{i_{(i)}})^T w_i^{(i)}}{d^T d} \tag{3.39}$$

Find

$$[err]_k^{(ik)} = \max\{[err]_k^{(i)}, 1 \le i \le M, i \ne i_1, ..., i \ne i_{k-1}\} \tag{3.40}$$

and select

$$w_k = w_k^{(ik)} = p_{ik} - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} w_j \tag{3.41}$$

where $\alpha_{jk} = \alpha_{jk}^{(iK)}, \quad 1 \le j \le k.$

The procedure is terminated at the M_s^{th} step when

$$1 - \sum_{j=1}^{M_s} [err]_j < \rho \tag{3.42}$$

where $0 < \rho < 1$ is a selected tolerance. A subset of M_s significant regressors is obtained.

3.5.2 Updating CSF Weights

The activation transfer function of the network is

$$a_{pj} = f_j(y_{pj}) \tag{3.43}$$

The sum of the square of the error for each input and output is given by

$$E_p = \frac{1}{2} \sum_j (d_{pj} - a_{pj})^2 \tag{3.44}$$

and the change in weight value according to EBP is

$$\Delta_P w_{ji} \propto -\frac{\partial E_p}{\partial w_{ji}} \tag{3.45}$$

The gradient component depends on the y_{pj} neuron, the error from the output of the jth neuron is contributed only by the weights w_{ji} for i= 1,2,....I for a fixed j. The derivative can be written as:

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial y_{pj}} \cdot \frac{\partial y_{pj}}{\partial w_{ji}} \tag{3.46}$$

The second term from equation 3.46 is the derivative of the dot product including centres, weights and distance function as in equation 3.12.

$$\frac{\partial y_{pj}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} [\sum_{k=1}^{n+1} (a_{pk} - c_{kj}) w_{kj} - \cos\omega_j \sqrt{\sum_{k=1}^{n+1} (a_{pk} - c_{kj})^2}] \tag{3.47}$$

The values a_{pi} and c_{ij}, for i=1,2,...I are constant for a fixed pattern at the input thus:

$$\frac{\partial y_{pj}}{\partial w_{ji}} = (a_{pi} - c_{ij}) \tag{3.48}$$

The error term at neuron j is defined as:

$$\delta_{pj} = -\frac{\partial E_p}{\partial y_{pj}} \tag{3.49}$$

The equations for the weight adjustment can be written using the error term δ:

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} (a_{pi} - c_{ij}) \tag{3.50}$$

$$\Delta_p w_{ji} = \eta \delta_{pj} (a_{pi} - c_{ij}) \tag{3.51}$$

Equation 3.49 can be re-written as

$$\delta_{pj} = -\frac{\partial E_p}{\partial y_{pj}} = -\frac{\partial E_p}{\partial a_{pj}} \cdot \frac{\partial a_{pj}}{\partial y_{ji}} \tag{3.52}$$

The last term in equation 3.52 shows change in output over change in input and is a first derivative of the activation function:

$$\frac{\partial a_{pj}}{\partial y_{ji}} = f'_j(y_{pj}) \tag{3.53}$$

To evaluate the first term in 3.52 two different cases are considered. For an output neuron of the network:

$$\frac{\partial E_p}{\partial a_{pj}} = -(d_{pj} - a_{pj}) \tag{3.54}$$

By substituting terms from equations 3.52, 3.53 and 3.54 the following expression is derived fro any output unit:

$$\delta_{pj} = (d_{pj} - a_{pj}) f'_j(y_{pj}) \tag{3.55}$$

The weight adjustment for neuron j in the hidden layer is proportional to the weighted sum of all δ values at the output layer of nosed connecting to neuron j with the output. The output layer error terms contribute to the adjustment of the hidden layer weights as:

$$\sum_k \frac{\partial E_p}{\partial y_{pk}} \cdot \frac{\partial y_{pk}}{\partial a_{pj}} = \sum_k \frac{\partial E_p}{\partial y_{pk}} \cdot \frac{\partial}{\partial a_{pj}} [\sum_i (a_{pi} - c_{ik}) w_{ik} - \cos\omega \| a - c_j \|] \tag{3.56}$$

and

$$\frac{\partial}{\partial a_{pj}} [\sum_i (a_{pi} - c_{ik}) w_{ik} - \cos\omega \| a - c_j \|] = w_{ik} - \cos \cdot \frac{a_i - c_i}{\| a_p - c_j \|} \tag{3.57}$$

$$\sum_k \frac{\partial E_p}{\partial y_{pk}} \cdot \frac{\partial y_{pk}}{\partial a_{pj}} = -\sum_k \delta_{pk} [w_{ik} - \cos \cdot \frac{a_i - c_i}{\| a - c_j \|}] = A \tag{3.58}$$

For the hidden layer neurons the δ terms in equation 3.52 can be defined as:

$$\delta_{pj} = f'_j(y_{pj}) \cdot A \tag{3.59}$$

From this the updated weights can be written as

$$\Delta_p w_{ji} = \eta \delta_{pj}(a_{pi} - c_{ij}) = \eta a_{pj}(1 - a_{pj})(d_{pj} - a_{pj})(a_{pi} - c_{ij}) \tag{3.60}$$

for output neurons

$$\Delta_p w_{ji} = \eta a_{pj}(1 - a_{pj}).A.(a_{pi} - c_{ij}) \tag{3.61}$$

also for hidden layer neurons that use the following activation transfer function

$$a_{pj} = \frac{1}{1 + \exp^{-y_{pj}}} \tag{3.62}$$

and its derivative

$$f'_j(y_{pj}) = \frac{\partial a_{pj}}{\partial y_{pj}} = a_{pj}(1 - a_{pj}) \tag{3.63}$$

The bipolar continuous activation transfer function can be used:

$$a_{pj} = \frac{2}{1 + \exp^{-y_{pj}}} = 1 \tag{3.64}$$

3.5.3 Updating CSF Centres

The centre adjustments are calculated using the same procedure as weight adjustments.

$$\Delta_P c_{ji} \propto -\frac{\partial E_p}{\partial c_{ji}} \tag{3.65}$$

From the chain rule, the derivative can be written as the change in error as a function with respect to the input and the change in input with respect to the centre values.

$$\frac{\partial E_p}{\partial c_{ji}} = \frac{\partial E_p}{\partial y_{pj}} \cdot \frac{\partial y_{pj}}{\partial c_{ij}} = -\delta_{pj} \cdot \frac{\partial y_{pj}}{\partial c_{ij}} \tag{3.66}$$

The last term in 3.66 can be determined as in equation 3.56 to form

$$\frac{\partial y_{pj}}{\partial c_{ij}} = -w_{ij} + \cos\omega \frac{a_i - c_i}{\|a - c_j\|} = B \tag{3.67}$$

The centre update for the hidden layer neurons can be written as:

$$\Delta_P c_{ji} \propto a_{pj}(1 - a_{pj}).A.B \tag{3.68}$$

3.5.4 Updating CSF Opening Angles

Similarly opening angle adjustment is governed by

$$\Delta_P \omega_j \propto -\frac{\partial E_p}{\partial \omega_j} \tag{3.69}$$

The derivative of the change in error with respect to opening angle is expressed as

$$\frac{\partial E_p}{\partial \omega_j} = \frac{\partial E_p}{\partial y_{pj}} \cdot \frac{\partial y_{pj}}{\partial \omega_j} = -\delta_{pj} \cdot \frac{\partial y_{pj}}{\partial \omega_j} \tag{3.70}$$

$$\frac{\partial y_{pj}}{\partial \omega_j} = \sin\omega \cdot_j \|a - c_j\| = C \tag{3.71}$$

The angle updates for the hidden layer neurons is given by

$$\Delta_P \omega_j \propto a_{pj}(1 - a_{pj}).A.C \tag{3.72}$$

CHAPTER 4

Radiograph Image Processing

4.1 Digital Image Processing

The principals of typical digital image processing cover a wide range of hardware use and software execution of a large variety of algorithms [36]. The first step in the process is *image acquisition*, this is the capture and digitisation of an image. In the case of radiographs image capture is done by way of an x-ray source directed at an object with a medium sensitive to x rays placed on the other side. Although up to date specialist mammography machines contain on board digitisation devices the mammograms from this study were taken before the availability of such machines and produced only radiographs on film. Thus creating the necessity for separate digitisation using specialised scanner devices for radiographs and further conversion into universal data types for software processing.

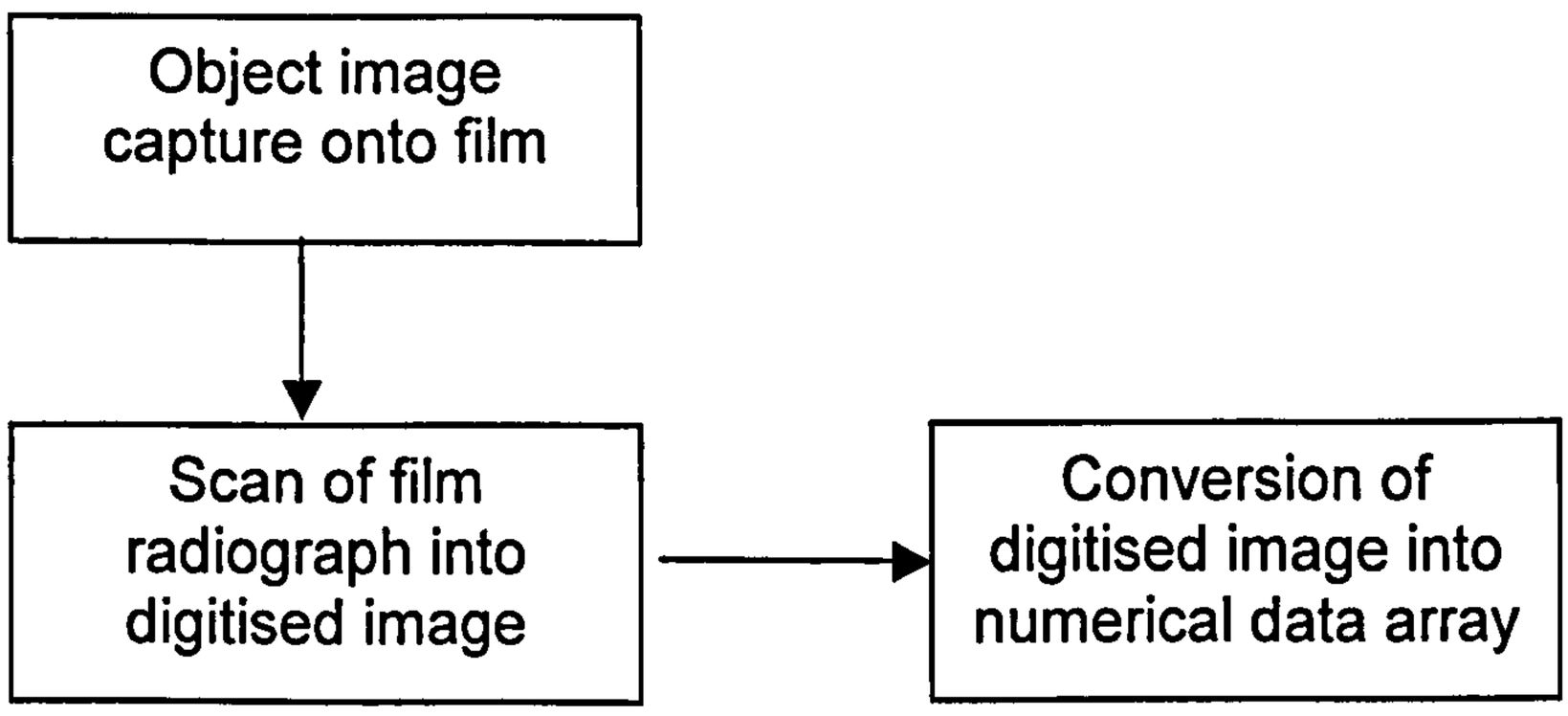


Figure. 4.1.1 Image Acquisition

4.2 Segmentation By Region Growing

Image Segmentation [36] is an image processing technique which deals with the analysis of the spatial content of an image. In particular, it is used to separate regions from the rest of the image, in order to recognize them as objects. Region Growing is an approach to image segmentation in which neighbouring pixels are examined and added to a region class if no edges are detected. This process is iterated for each boundary pixel in the region. Advantages of region growing are that the borders of regions found by region growing are perfectly thin (since we only add pixels to the exterior of our region) and connected. Membership in a region can be based on multiple criteria in the iteration process. However the main disadvantage to region growing is it is computationally expensive. It takes both extensive computing power (processing power and memory usage) and time to implement the algorithms especially in the case of large images.

4.2.1 Stack Based Region Growing Algorithm

Region growing is a classical heuristic method for the separation of region within an image also known as *flood filling* [36]. Once a region that is required to be separated from the image is identified, the characteristics that are common among all pixels in the selected region are established. For radiographs pixel quality are defined within a grey scale. In general region growing is an iterative pixel by pixel approach that categorises each pixel as being in the region of interest or outside. Using a stack to store pixel co-ordinates that have been selected as within the region of interest so that unnecessary repetition of testing pixels that have already been categorised.

This stack based approach is implemented as follows:

1. The first stage is to identify the co-ordinates of one pixel that is within the region of interest by inspection, this pixel is labelled as the *seed* pixel and is placed at the bottom of the stack.

2. The co-ordinates of the eight pixels that surround the seed are identified and the predetermined condition (for example if the pixel's grey level higher than a threshold or not) is tested. All eight neighbouring pixels are tested and only placed in the stack on top of the original seed if the result of the condition is positive and if this pixel is not already in the stack.

3. After the seed's neighbours are tested the algorithm checks to see if it required to continue by checking if the seed co-ordinates are at the top of the stack. The pixel that is labelled as the seed changes if this is not the case and the seed becomes the next pixel up in the stack and step 2 is re-iterated.

4. When the neighbours of all the pixels in the stack have been tested and the seed pixel rises to the top the process stops. All pixels that have given a positive outcome to the condition have been accounted for in the stack the surrounding pixels of these stacked co-ordinates are negative, thus one region of a common condition has been identified.

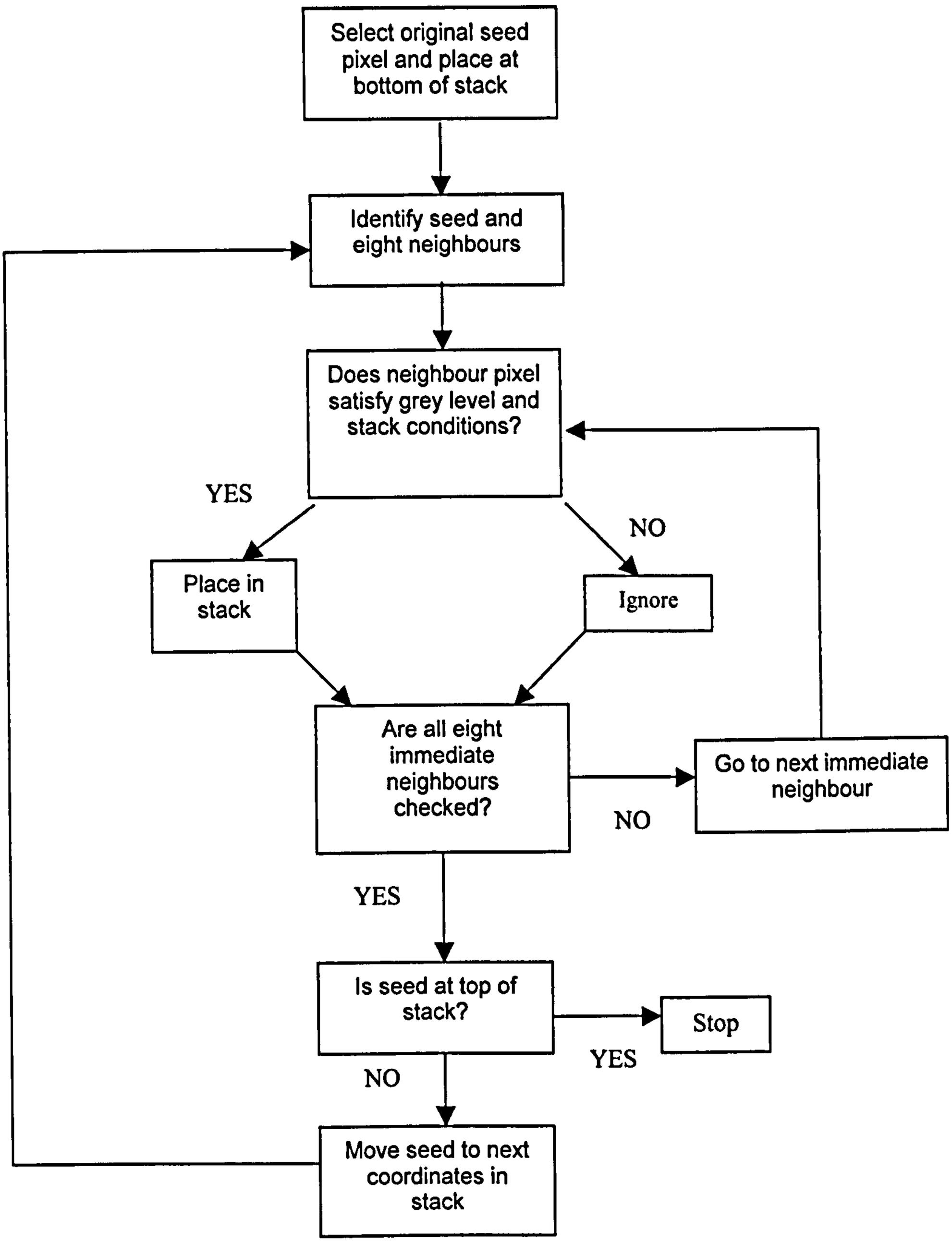


Figure. 4.1.2 Stack Based Region Growing Algorithm

4.3 Noise Reduction With Neighbourhood Averaging

Noise reduction is an important part of image processing. In this case where the digitised images are created as second generation image, in other words the images were not created directly from the object but were scanned from the radiograph, noise reduction is especially important. The scanning process can give rise to noisy images which can have a large impact on sensitive algorithms such as region growing.

The neighbourhood averaging method or *smoothing* takes into account each pixel and its nearest neighbours. In a one dimensional case the process finds the mean of the grey levels of a set of that are within a maximum distance N from the subject pixel. Then sets the subject pixel's grey level to that mean value. The process then moves to the next pixel until the end of the image row is reached. This does have the effect of blurring the image more prominently if the distance N is large. For am image of dimensions y rows and x columns then for row y the process applies the function:

$$g_{new}(x) \Rightarrow \frac{[g(x-N)....g(x-1)+g(x)+g(x+1)...g(x+N)]}{N+1} \tag{4.1}$$

Where g represents the grey level of the pixel at co-ordinates (x,y). This process can be repeated for each row. One flaw in this process however is the averaging of the pixels that are within N distance of the image edge. The process has no pixels past the edge to evaluate a true mean with. This can be over come by either buffering with fake pixels of grey levels similar to that of the pixel at the edge or buffering with fake pixels with a zero value.

4.4 Image Enhancement High Pass Filtering

Processing of radiographs often require improvement of image quality. In the case of mammograms where boundaries and regions of high contrast are of interest enhancement is of these areas is valuable. By transforming the image into the frequency domain using Fourier and Inverse Fourier transforms [36] the image can be powerfully manipulated using special filters. Defining the spatial image as *f(x,y)* the Fourier transform of the image is defined as:

$$F(u,v) = \int_{-\infty}^{\infty}\int f(x,y)\exp[-j2\pi(ux+vy)]dxdy \tag{4.2}$$

Where u and v are known as the frequency variables and j represents the component as complex. The frequency domain function of an enhanced image *G(u,v)* can be yielded if a transfer function *H(u,v)* is used.

$$G(u,v) = H(u,v)F(u,v) \tag{4.3}$$

Enhancement of high contrast components can be achieved using *high pass* filters. In the frequency domain an ideal high pass filter returns *H(u,v)* as 1 if u and v in the frequency plane gives a point that is more than a threshold value D_o. The filter returns 0 if the point falls below the threshold. In a two dimensional frequency plane D_o can forms a circular boundary about the origin.

This distance from the origin is defined as

$$D(u,v) = \sqrt{(u^2 + v^2)} \tag{4.4}$$

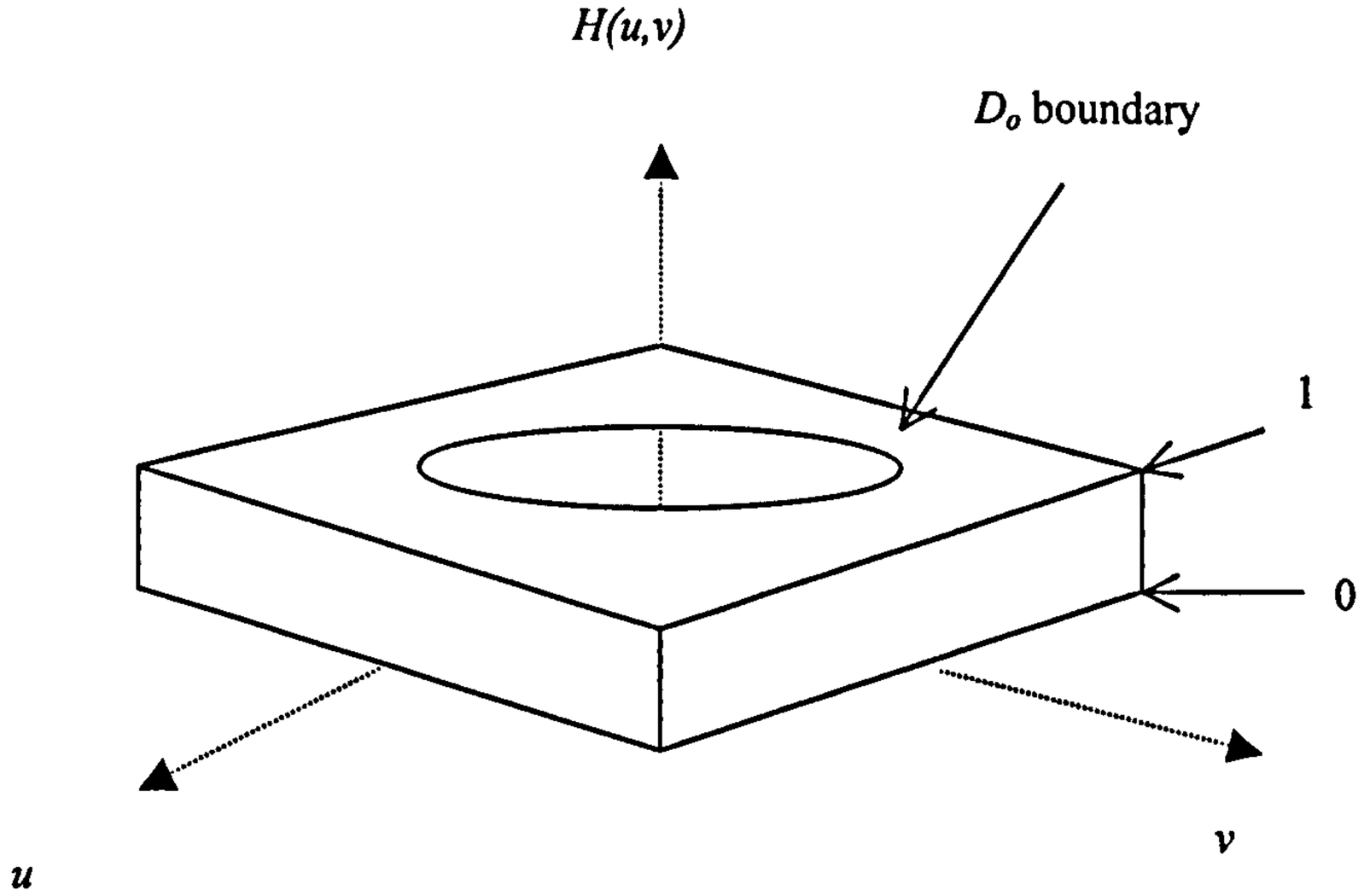


Figure. 4.4.1 Ideal High Pass Filter Frequency Domain

The filter is defined as

$$H(u,v) = \frac{1}{1+[D_o / D(u,v)]^{2n}} \tag{4.5}$$

Where n is the known as the order of the filter.

CHAPTER 5

Mammogram Asymmetry Analysis

5.1 Symmetry Features

Previous studies [83] have shown some correlation between estimated mammogram pair volume difference and the eventual occurrence of malignant bodies and cancer. This study calculates mammogram pair area difference from digitised images, as a measure of asymmetry. For this measurement the known breast tissue region must be separated from the image then further calculation of the area of this region. The difference between the area of the left and right image is scaled by considering that sum of the two areas. This is so that a more true reflection of the degree of asymmetry is gained. All mammograms are of *cranio-caudal* view and acquired with the chest plate region of the subject on the left.

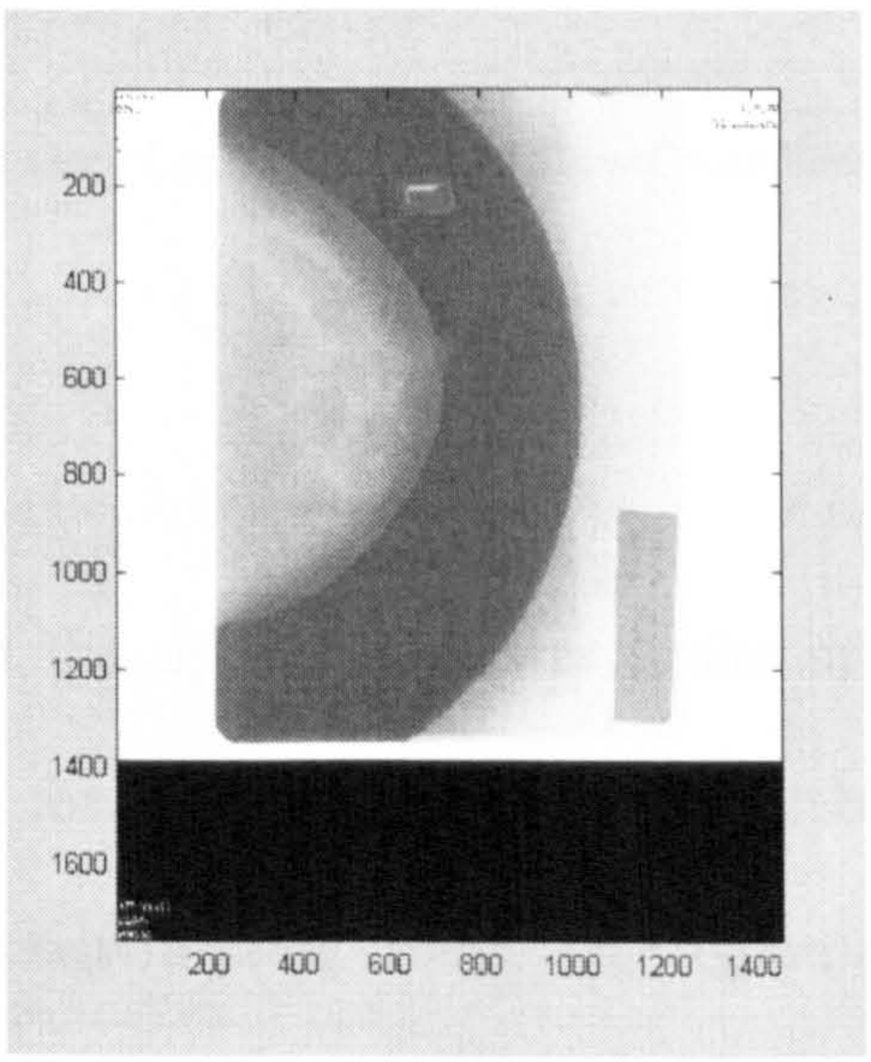


Figure. 5.1.1 Unaltered Cranio Caudal Image

5.2 Segmentation Results

The grey scale for pixels is defined numerically as 255 for white and 0 for black. By inspecting the variation in grey level for a medium row within the image that contains tissue regions as well as non tissue regions the following characteristic is found in figure 5.2.1.

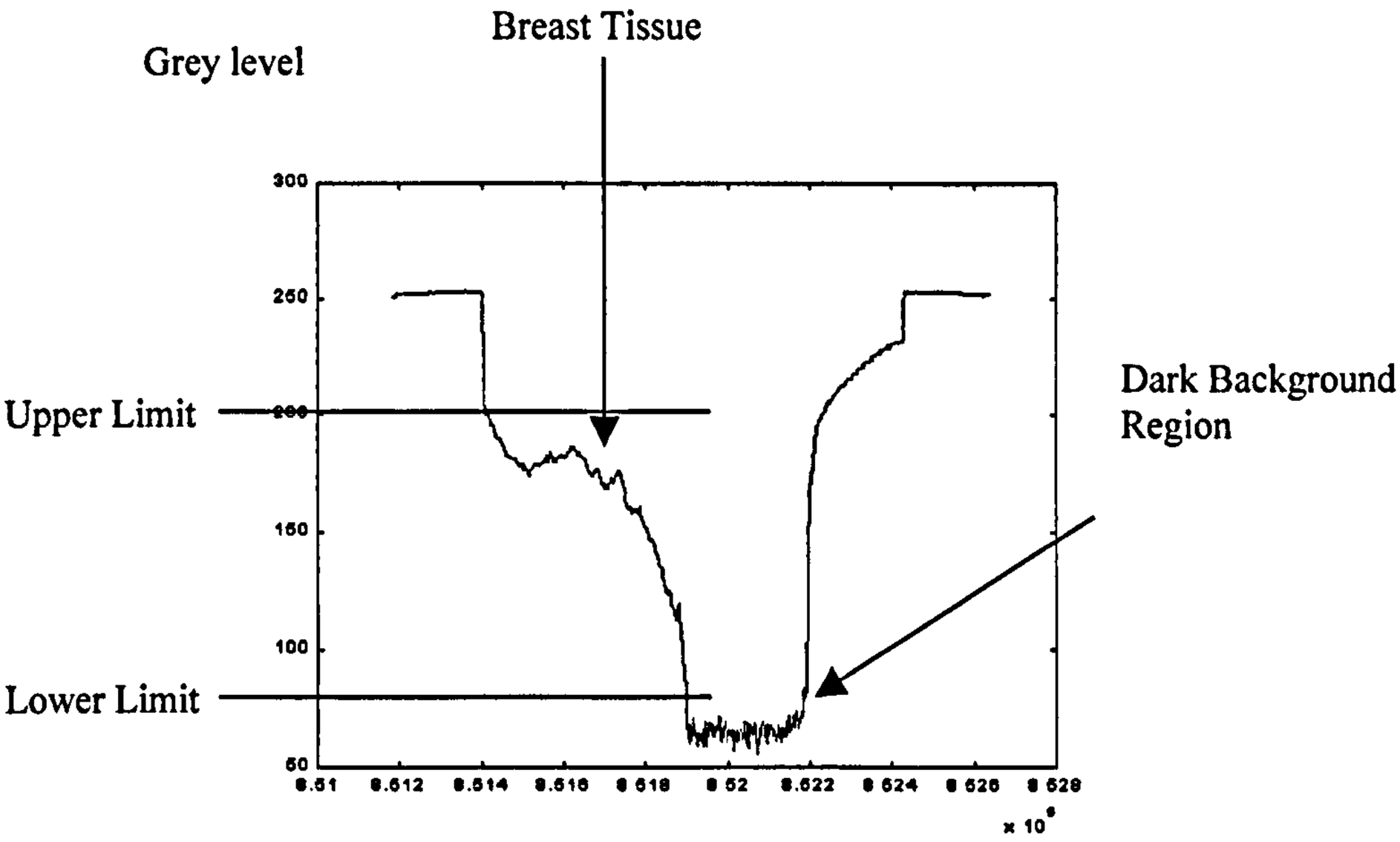


Figure. 5.2.1 Grey Level Variation of Row Cross Section

From this cross section the upper and lower limits of the tissue region can be determined. The criteria for region growing can be established as the grey level of each tested pixel falling within these limits. The region growing process beginning with an arbitrary pixel central within the tissue area.

A common problem that occurred in many images after the application of region growing is neighbouring non tissue pixels with similar grey levels caused the growing region to 'leak out' and process border areas with similar grey levels to be misclassified.

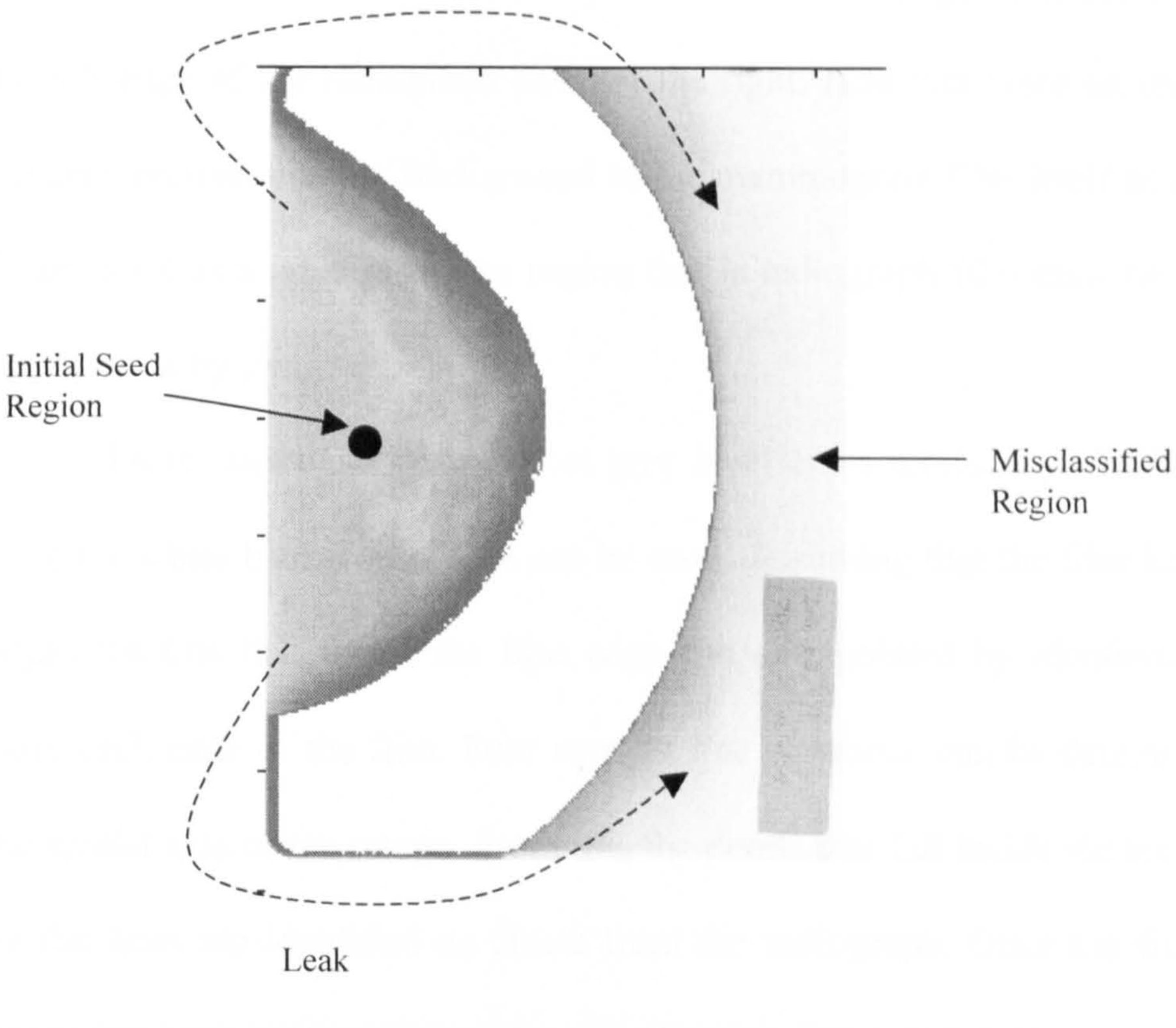


Figure. 5.2.2 Image of an Unsuccessful Grown Region

Moreover this method known to be time consuming proved to be highly inefficient as large misclassified regions were tested. Although the stack based approach inherently counts the number of pixels for the measurement of tissue area due to the leaking effect this process is unsatisfactory.

5.2.1 Image Specific Segmentation

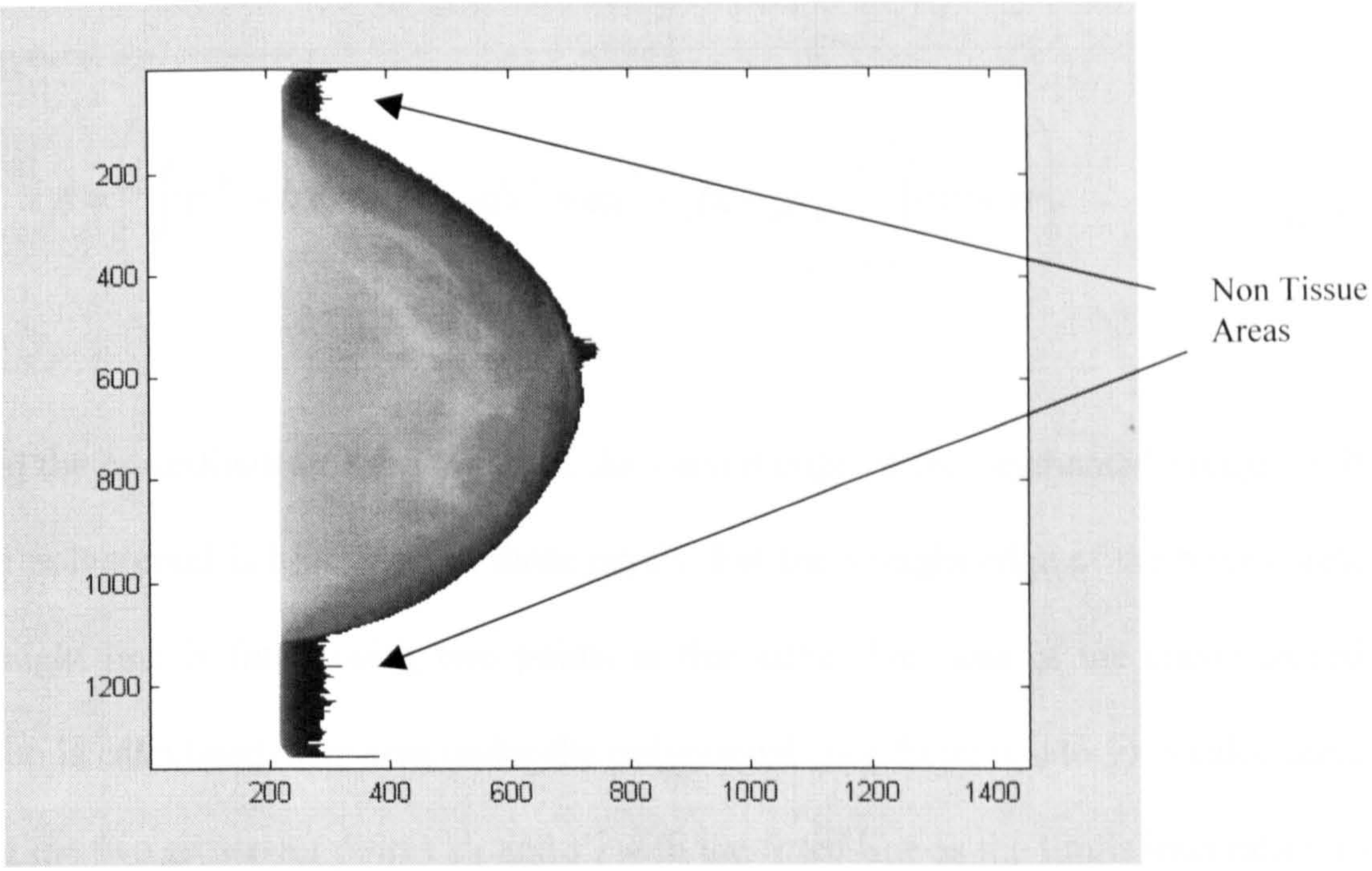
Alternative methods for segmentation specific to the mammograms with the characteristics shown in figure 5.2.1 lose generality. However gains faster computation time and successful segmentation. One such algorithm is a row by row iteration that is based on the known fact that all the images contain tissue region on the left edge of the radiograph and not the right. However since an unaltered image contains regions that are background to the mammogram film itself as can be seen in figure 5.1.1 as a white area, the region that is radiograph film must be selected. This can be done by *image cropping*.

From horizontal and vertical grey level cross sections of the image the drop from the white background area can be seen. Assuming that the film has straight line edges the line that forms the film edge can extrapolated by identifying two points from each edge of the film. Four straight line equations can be determined based on the spatial axis of the image. From this the pixels that fall inside the rectangle formed by the lines are identified as pixels from the radiograph. Once the film area of the image has been cropped tissue area segmentation can commence.

Beginning with the highest row the pixels from left to right are tested see if the grey levels fall below the lower limit. This will include the leaking regions that do not fall below the lower limit however these regions can be eliminated from the area calculation later in the process.

This image specific method proved to be less time consuming and computationally intensive. The large misclassified area created by region growing is ignored due to the emphasis on the left sided pixels in this algorithm.

Figure. 5.2.3 Image of Segmented Region



5.3 Area Calculation

One drawback that the image specific segmentation produced as compared to region growing as the necessity to evaluate the area of the tissue region additionally. By fitting a curve to the right edge of the segmented image and by considering the left vertical straight line gained from cropping, the area between these lines can be calculated. The equation of the straight line is known however using standard regression methods a curve can be best fit to the set of pixels that make up right edge of the tissue region. The co-ordinates of these pixels are known from the row by row segmentation. The area of the space between the two lines and be calculated from subtraction of the area under the right edge curve and the area under the left edge line.

These areas individually can be evaluated using integration as follows:

$$A=\left(\int_{x_1'}^{x_2'} ax^6+bx^5+cx^4+dx^3+ex^2+fx+g\right)-\left(\int_{x_1'}^{x_2'} mx+n\right) \tag{5.1}$$

Using the co-ordinates of the pixels at the curved edge of the segmented image a 6th order polynomial is best fitted to those pixels. For the straight edge of the breast area, a straight line is fitted using two points at that edge. The area of the encompassed section is calculated. The area under the polynomial (coefficients a to g) is calculated, using the two crossover points x'_1 and x'_2 with the fitted line as the limits integration in equation 5.1.

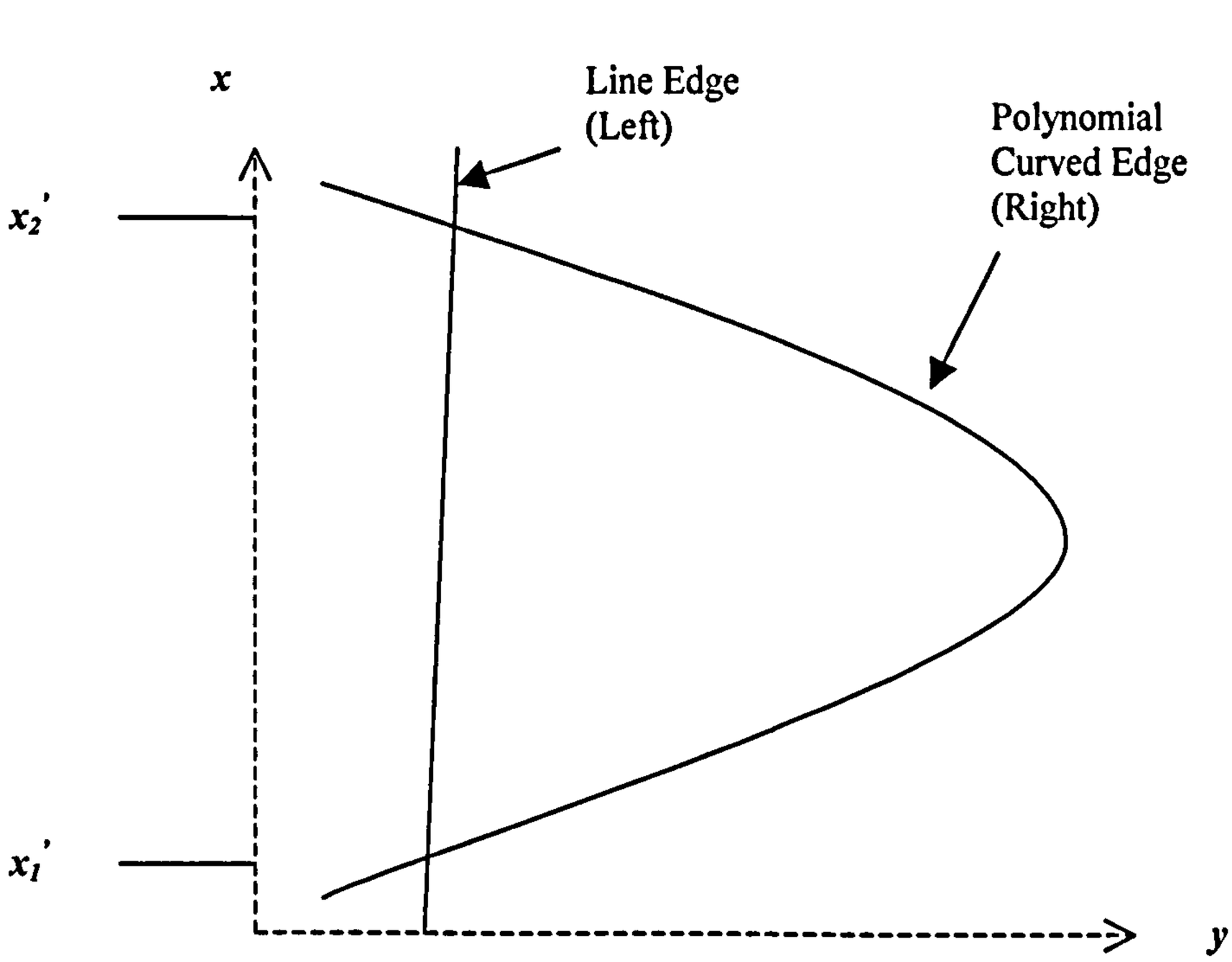


Figure. 5.3.1 Best fit boundaries

5.3.1 Shape Similarity Measure

In addition to the difference in left and right areas of the mammogram pairs, similarity between the curves for a subject is measured as further aspect of asymmetry. The measure in this study is a simple morphological translation of 100 points from one polynomial to another. In the case of the images in this study the variant attribute is represented by the relative positions between the curved and straight edge of the segmented image. All of the images have crossover points (used as limits of integration in part 1). The vertical distance between these points is used to identify 100 equally spaced sample points (including the crossover points). This algorithm starts by mapping the lowest of these points to the origin (both curves). The Euclidean distance between the remaining points on both curves can then be calculated. The summation of these distances gives an overall measure.

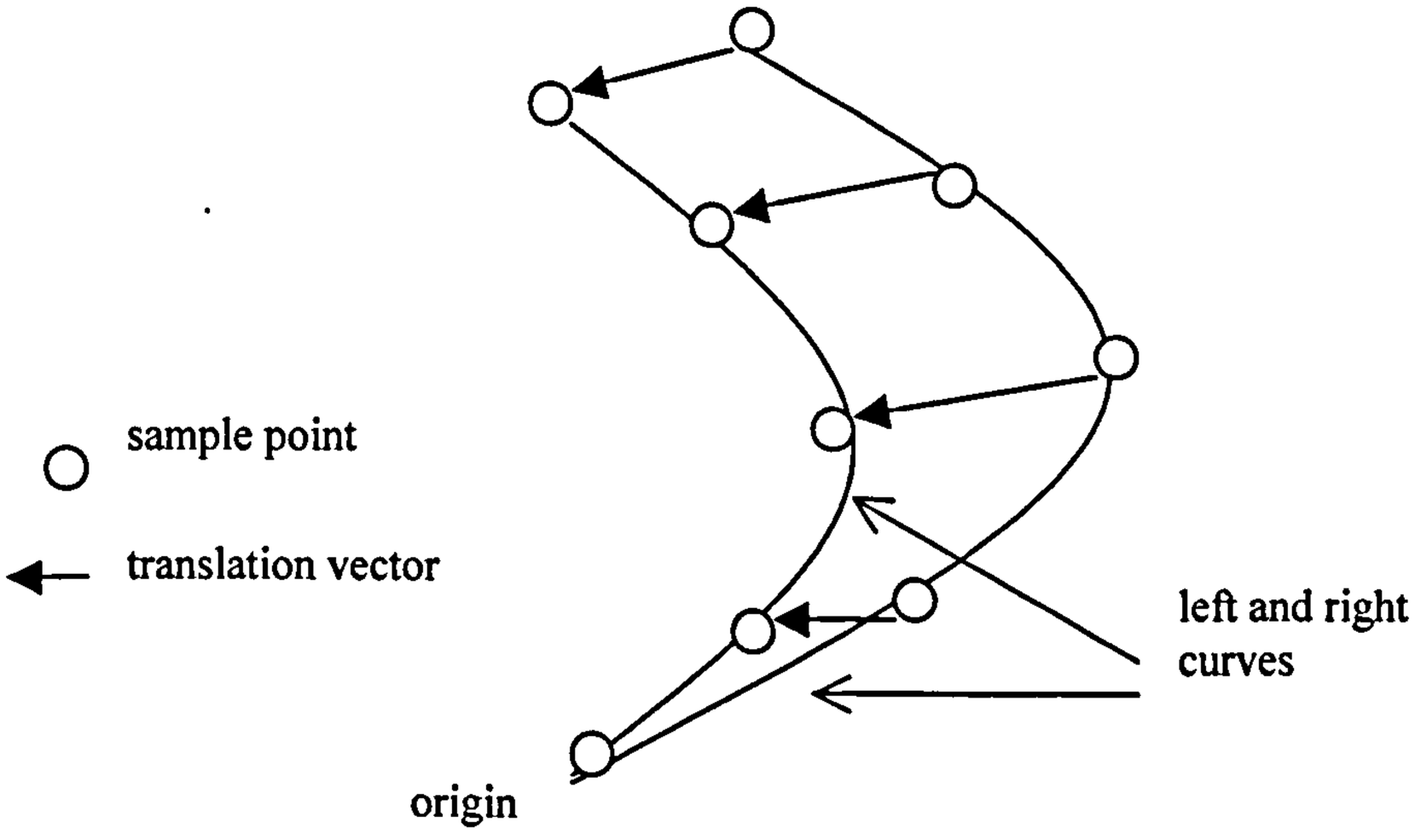


Figure. 5.3.2 Demonstration of Translations for Sampled Points

However this is not a pure representation of shape as it is influenced by the curve size. Using the maximum widths, each curve is scaled (maximum equals 1) in the horizontal direction. Scaling in the vertical direction is done by simply discounting the vertical element of the translation vector, this assumes all the sample points are of equal vertical interval. The summation of the horizontal differences of the 100 points is then calculated.

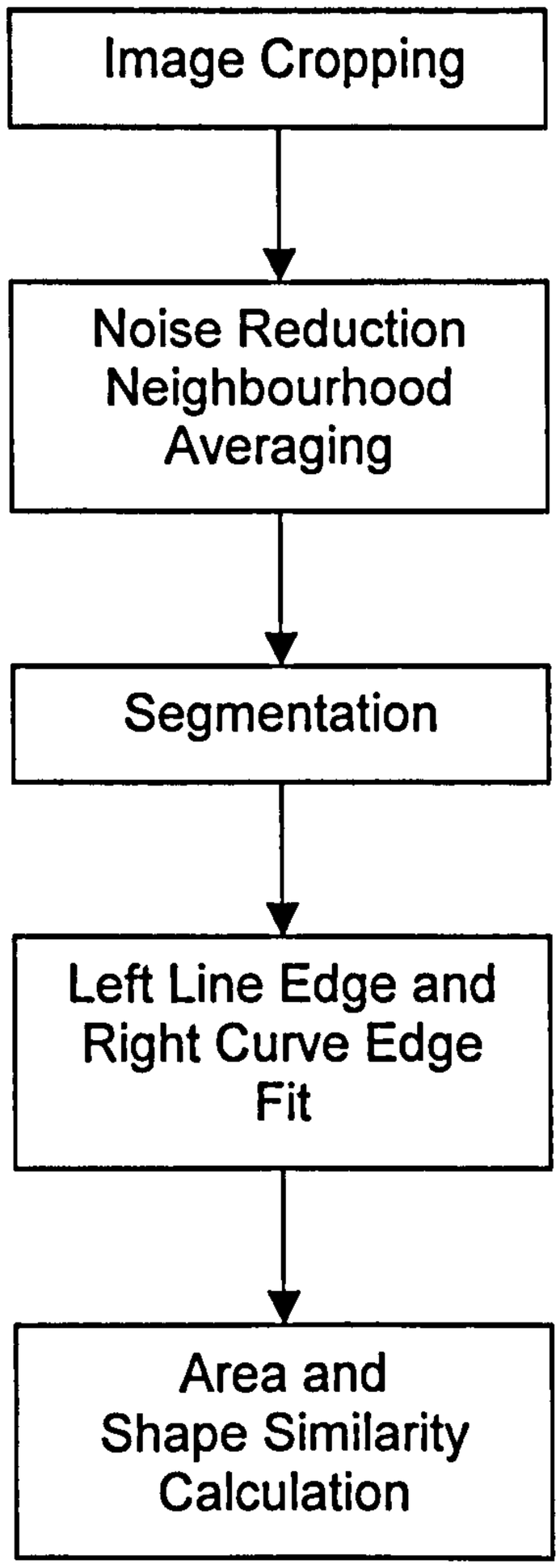
Two measures are obtained:

1. Non-scaled shape differences between curve pairs. (size dependant).
2. Scaled shape differences between curve pairs. (size independent).

5.4 Asymmetry Feature Extraction Process

Many of the digitiesed mammogram images were found to have a high degree of noise causing the segementation process to yield an uneven edge on the right curved side. Reduction of noise was acheived using neighbourhood averaging with small integer values of neibourhood size *N* depending on the degree of noise. After which segmentation and the measure of size difference and shape for Asymmetry feature extraction is applied.

Figure 5.4.1 Asymmetry Feature Extraction Process



5.5 Pattern Feature Extraction

An additional feature obtainable form the digitised mammogram images is the texture of the tissue region. However such radiograph images have highly complex subtle tissue pattern structures that reduction to the most prominent features is required. The high contrast regions apart from tissue to non-tissue boundaries are formed by a high gradient of tissue density change within the subject. This manifests as white 'whisp'

like patterns in the mammogram. A high pass filter used to emphasise high contrast regions is required with further polarising of grey levels using a threshold value. All values of pixels with grey levels under the threshold are made to 255 to produce an image containing only the desired pattern regions.

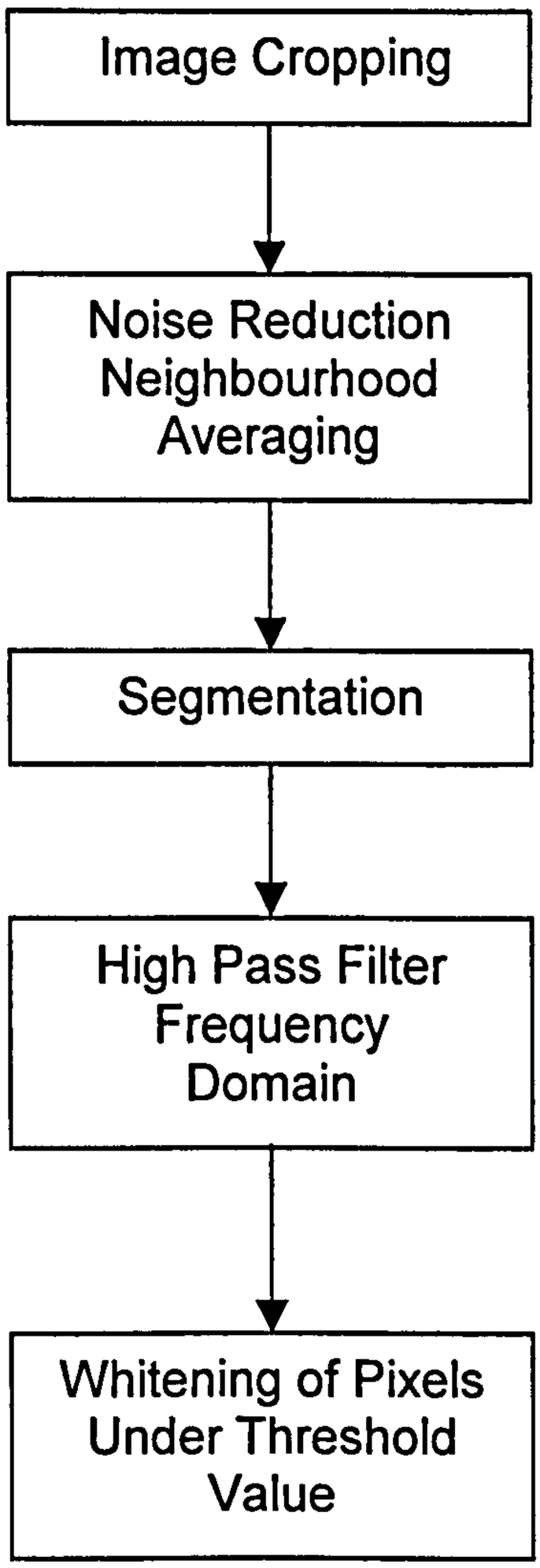


Figure 5.5.1 Pattern Feature Extraction Process

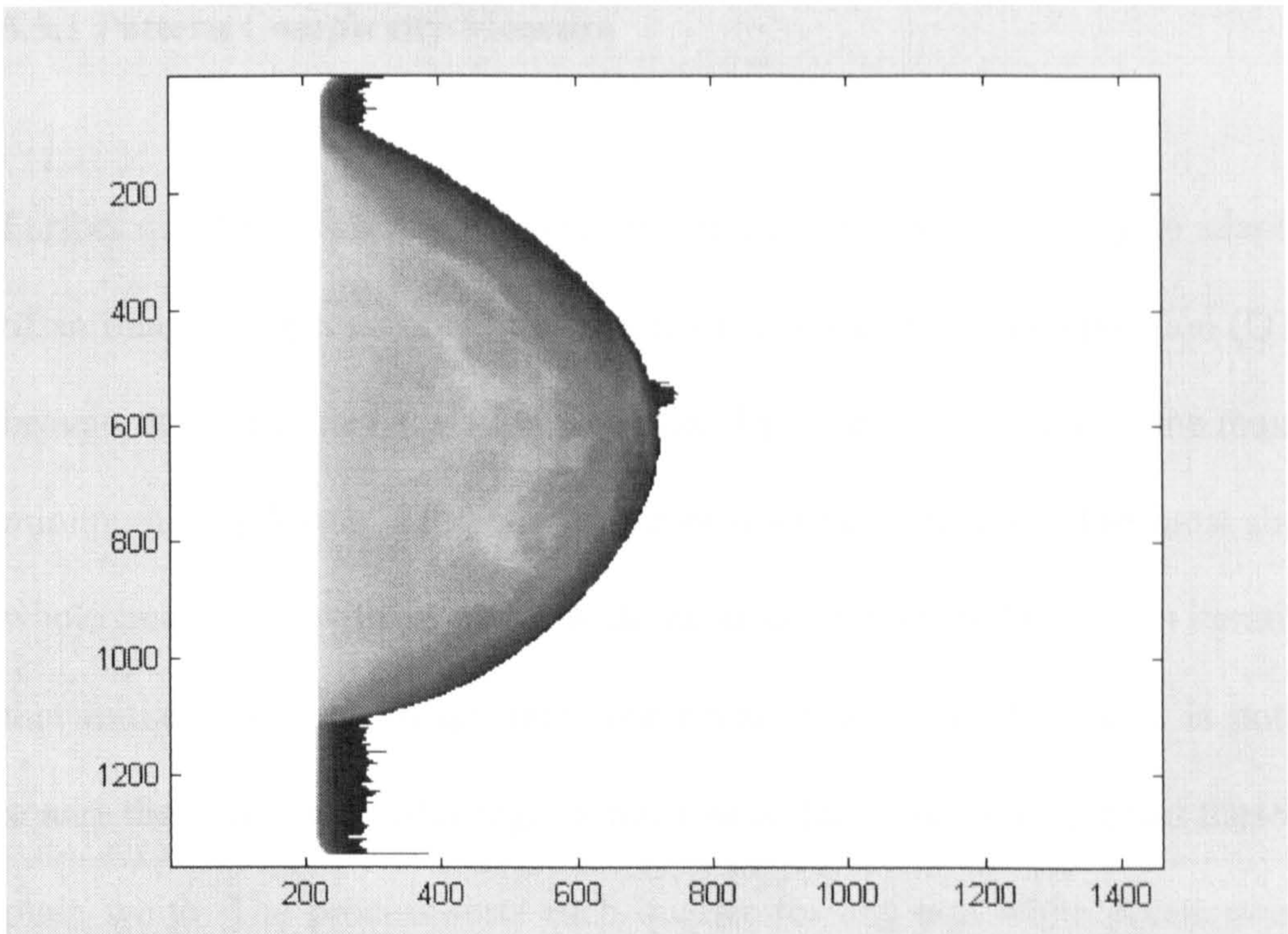


Figure 5.5.2 Segmented Only Image

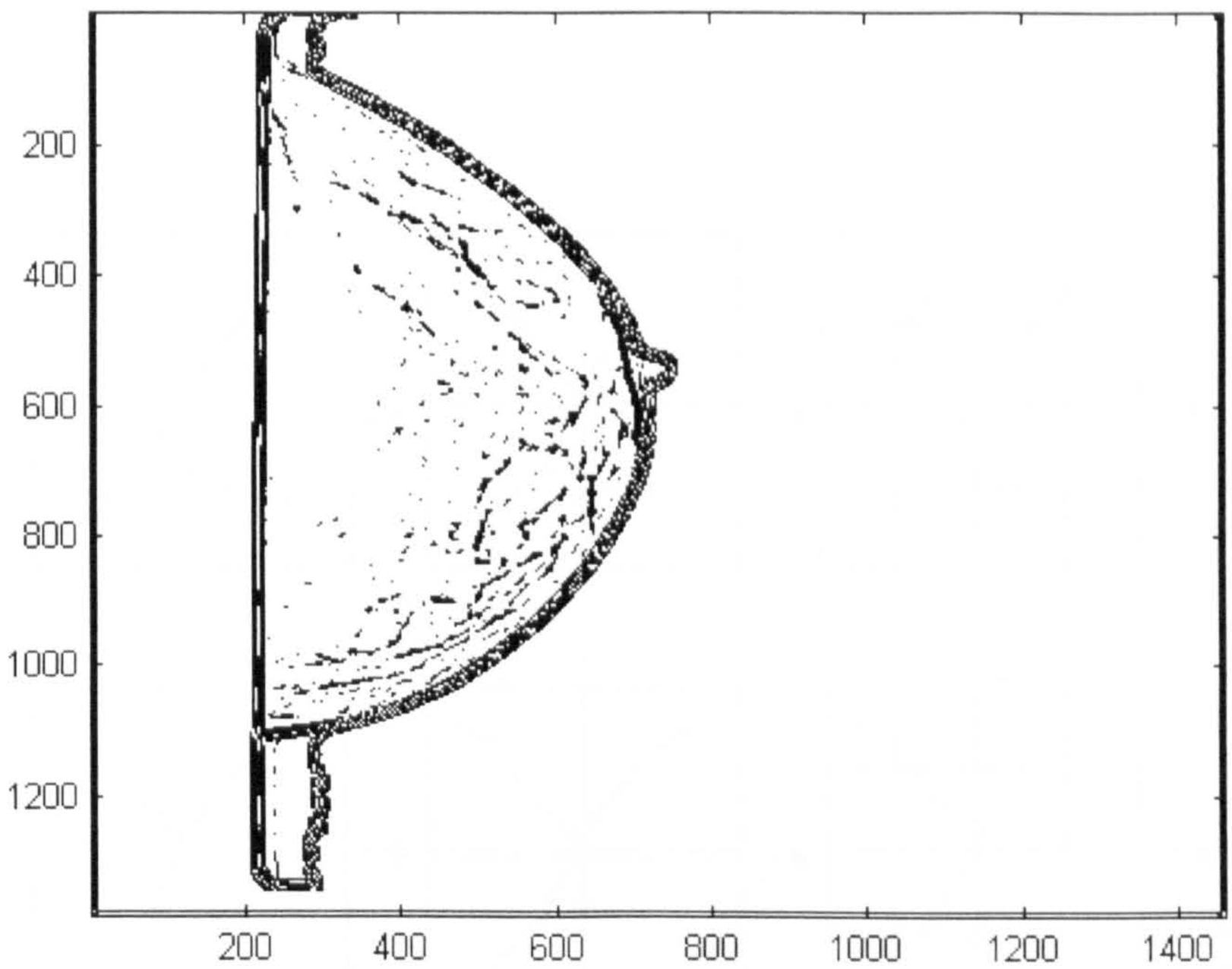


Figure 5.5.3 Filtered and Binarised Image

5.5.1 Pattern Complexity Measure

Further quantification of the filtered image can be processed using an adapted version of an image compression algorithm known as quad tree decomposition (QTD). It can be said that for a grey scale image a pixel by pixel 'chessboard' of the maximum and minimum grey levels is the most contrasted and complicated. The most simple is the whole image area with all pixels with the same grey level. QTD is an iterative process that initially cuts the image into four equal quarters. If the image is not originally square then minimal buffer regions must be added. It is important that these region are plain white. The process tests each quarter for any non white pixels present in the region. If this criterion returns positive then QTD cuts that quarter into a further four quarters. This process continues as far as required ending at single pixel level if need be.

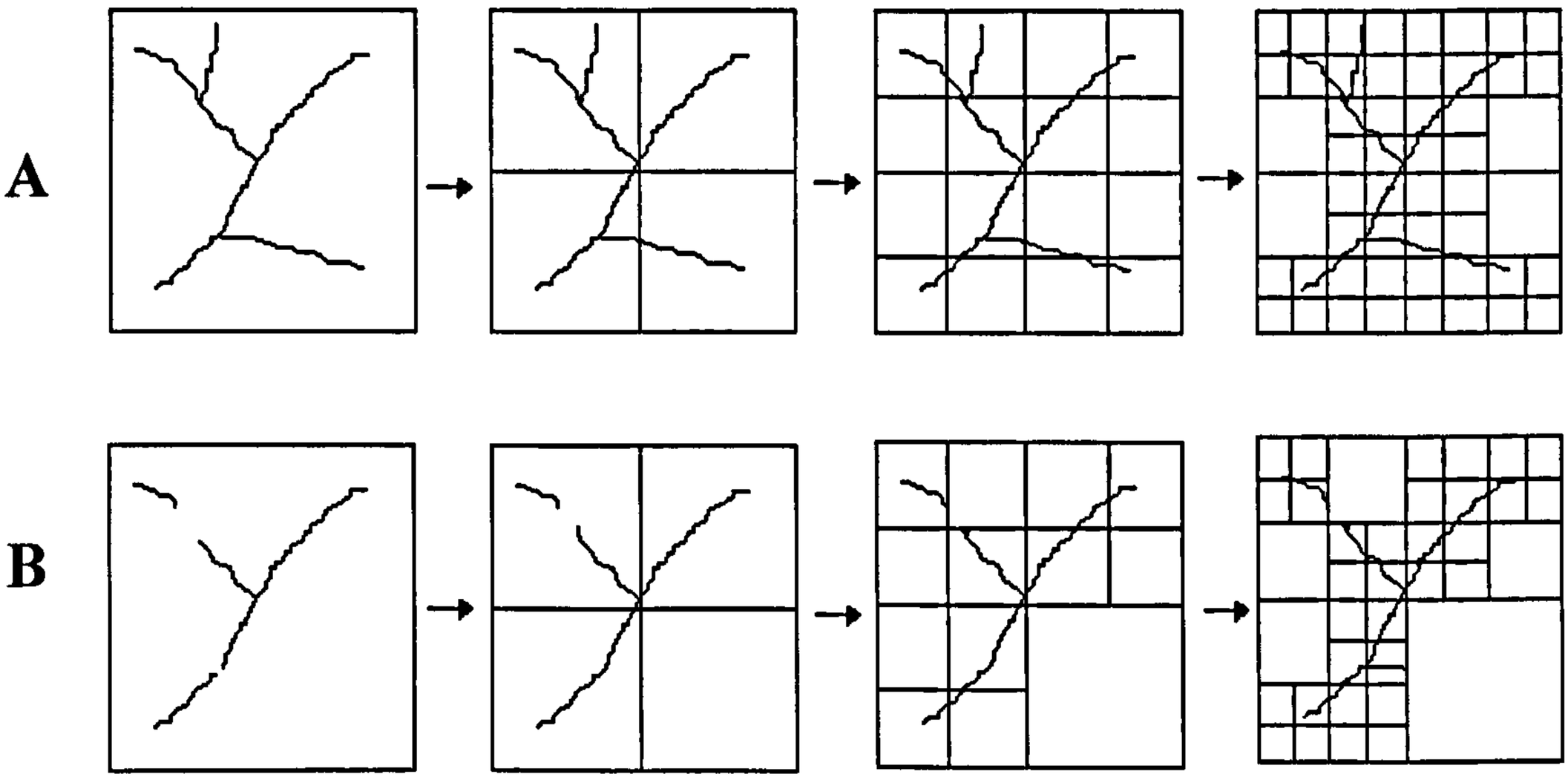


Figure 5.5.4 Demonstration of QTD process

Figure 5.5.4 shows two similar images 'A' and 'B', image 'A' contains more branches and is a more complex pattern. QTD applied to both images returns more small quarters present in the last diagram for example 'A' than example 'B'. Both images return 4 four large quarters and again 'A' returns more medium quarters. However for larger images such as filtered mammograms this process returns quarters ranging in size from a quarter of the original buffered image size to a single pixel. Images that return a high number of small quarters contain more complex non background patterns. The disadvantage of QTD is that the process is very time consuming for square images with pixels numbers at 1400^2 such as these.

CHAPTER 6

Data Interpretation and Neural Network Simulation

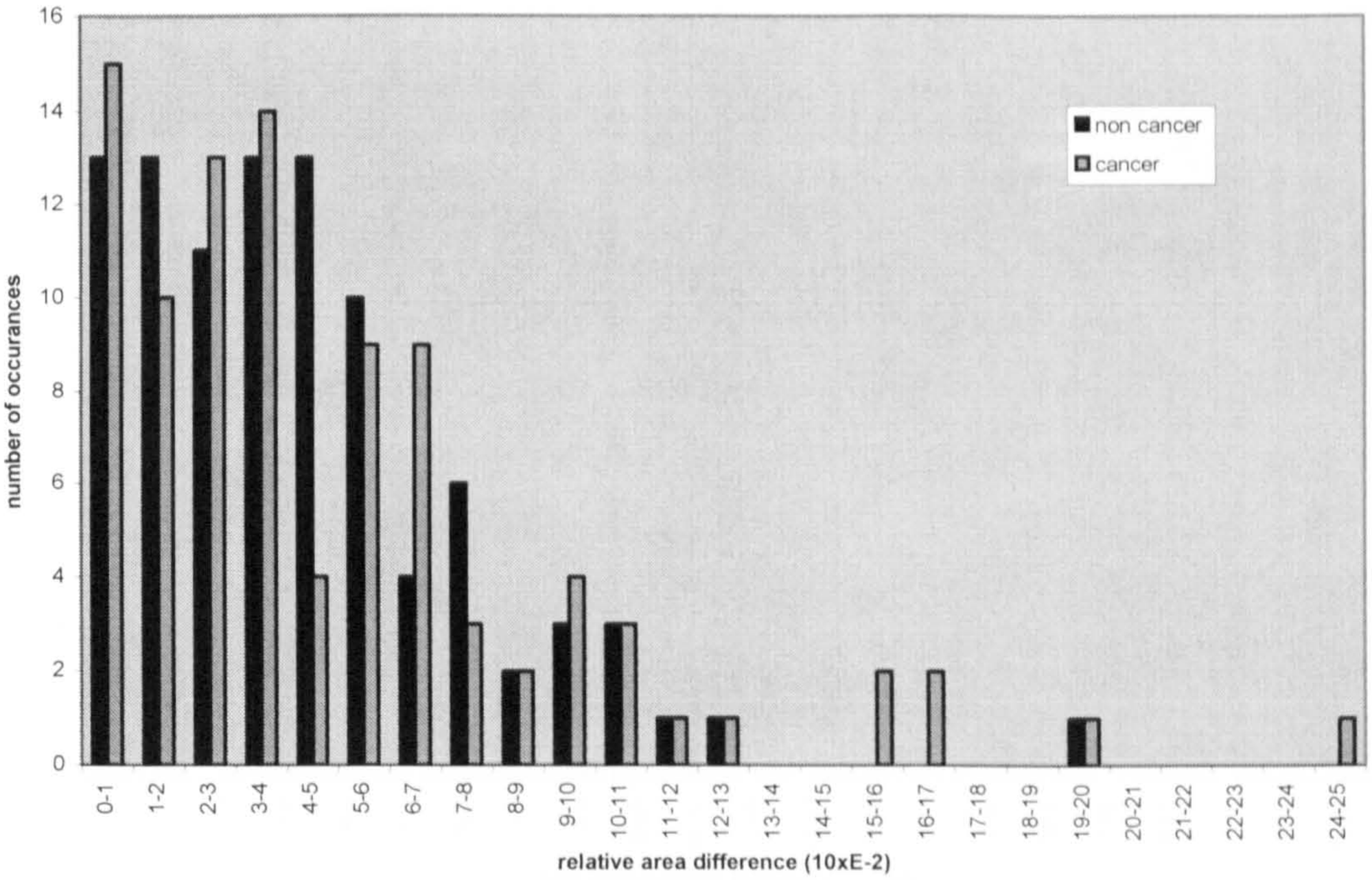
6.1 Asymmetry Calculation Results

In accordance with principles on cancer pre-dispositions [56], high asymmetry or fluctuating asymmetry (FA) [83] is more prevalent in the cancer sets. The comparison of this prevalence between the three different forms of FA can be made. The most notable being the non scaled difference for shape similarity evaluated using the method in section 5.3.1. The contrast between the cancer and non-cancer sets for scaled difference is not as high as the features that are influenced by breast size (relative area and non scaled difference).

High symmetry is also prevalent in all cases. However the attributes influenced by breast shape (scaled and non-scaled difference) does produce a minimal level of asymmetry where few (or none in the case of scaled difference) fall under. This is not found in relative area difference.

The following histograms present the results for a test batch of approximately 200 pairs of mammograms. The number of occurrences is plotted against intervals of ascending FA with the lower limit inclusive to the interval. *Scutt et al* [83] demonstrated pre-disposition trends using difference in volume. The measures in the figures below can be compared to the trend shown in figure 1.4.1.

Figure 6.1.1 Relative Area Difference Comparison



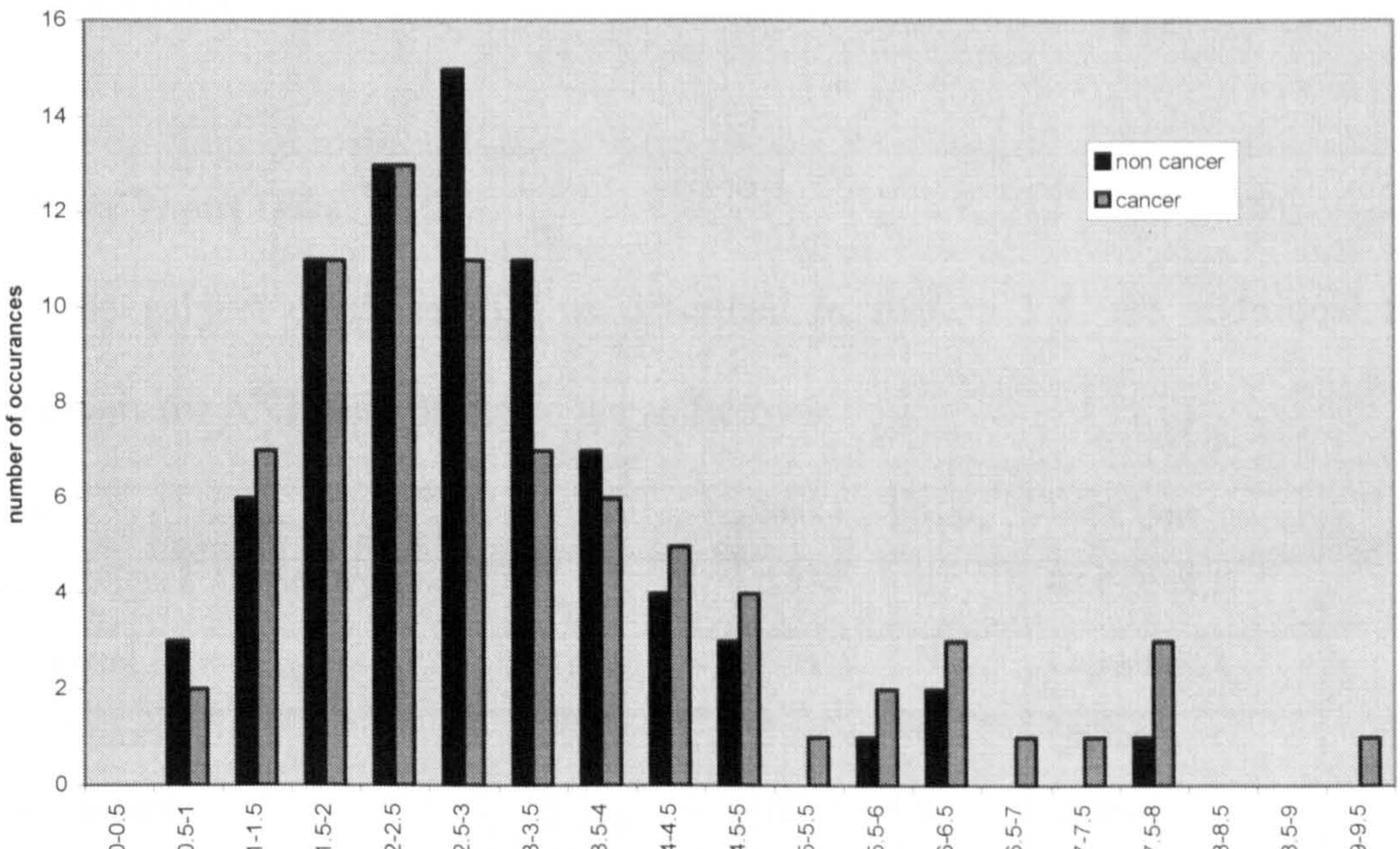
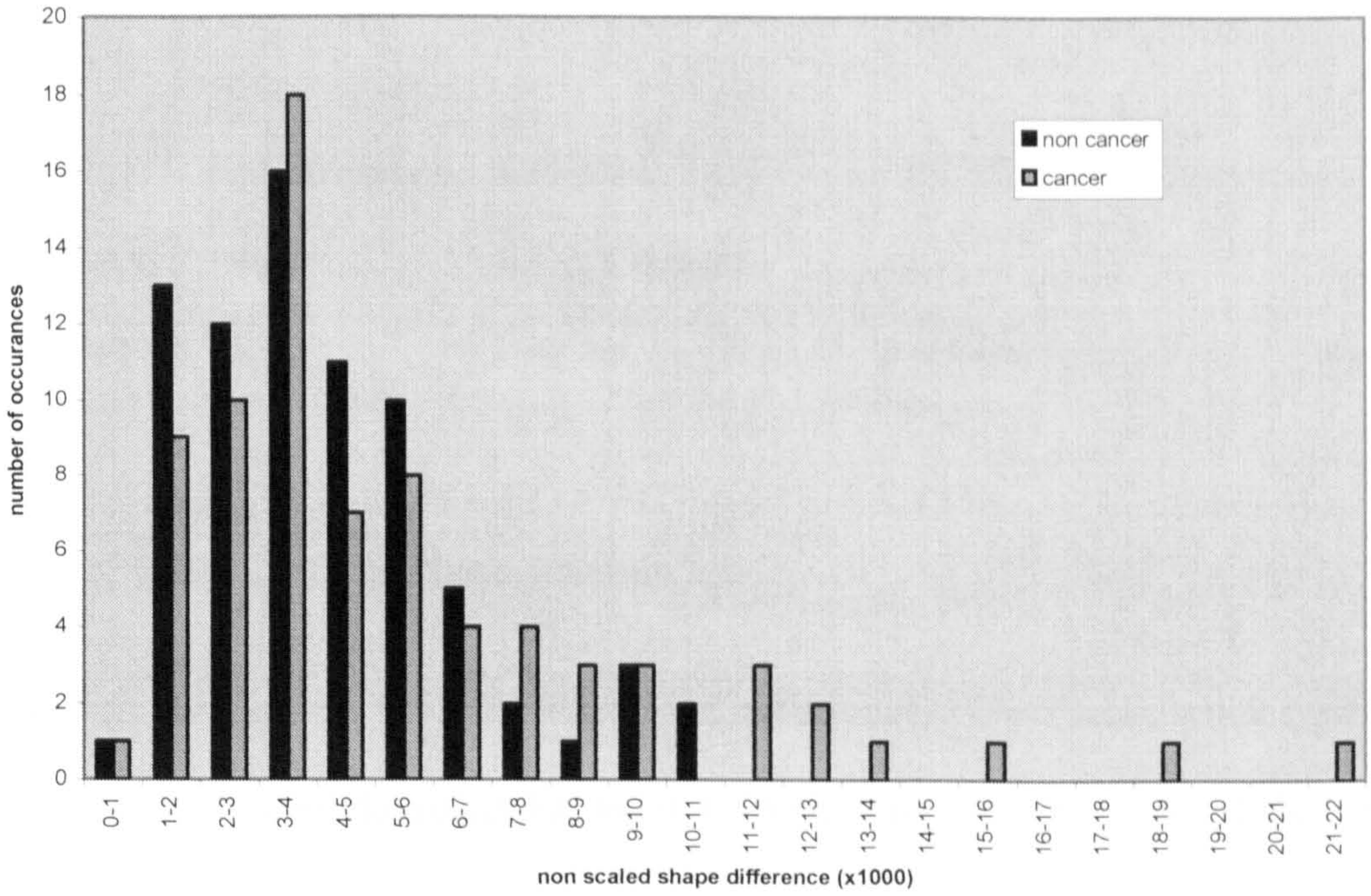


Figure 6.1.2 Scaled Shape Difference Comparison

Figure 6.1.3 Non-Scaled Shape Difference Comparison



Asymmetry calculation constitutes one input dimension for training feed forward neural networks.

6.2 Other Input Data

Form the subject database [83] as described in section 1.5, the additional input dimensions for ANN interpretation are as follows:

Feature	Max	Min	Data Type
Relative Volume Asymmetry (Quotient)	0.438	0	Real (max 1)
Parenchymal Type	N/A	N/A	Categorised
Family History	5	0	Ordinal
Age at Menarche	18	9	Ordinal
Pill Use	1	0	Binary

Table 6.2.1 Input Dimensions

With the exception of the relative asymmetry the all of the input data is discrete. The two unbounded input factors are categorised as follows:

Data	No of intervals	Definition
Family History	4	0, 1, 2, 3 or more
Age at Menarche	6	11 or less, 12, 13, 14, 15, 16 or more

Table 6.2.2 Ordinal Input Definitions

These were categorised from the number of occurrences. The Parenchymal type does not need to be further classified. However both the left and right breast of the subject must be considered individually.

A final dimension for use as an input is related to high levels of Oestrogen exposure as stated in section 1.5, this factor is most relevant to use or non use of the contraceptive pill by the subject. This translates to a binary input for ANN simulation, 1 for use and 0 for non-use.

Although the subject database contained 898 cases it was found that 564 cases had all of the required factors available. Out of which approximately 400 of the 564 had corresponding pairs of mammograms available and in tact. Due to the clinical use of many of the mammograms since as early as 1978 many of the mammograms from the archives had been damaged, altered or missing.

Input Neuron	Representation
1	relative area asymmetry
2	no family members diagnosed
3	one family member diagnosed
4	two family members diagnosed
5	three or more family members diagnosed
6	left parenchyma type is class N1
7	left parenchyma type is class P1
8	left parenchyma type is class P2
9	left parenchyma type is class DY
10	right parenchyma type is class N1
11	right parenchyma type is class P1
12	right parenchyma type is class P2
13	right parenchyma type is class DY
14	menarche age is 11 or less
15	menarche age is 12
16	menarche age is 13
17	menarche age is 14
18	menarche age is 15
19	menarche age is 16 or more
20	contraceptive pill use (binary)

Table 6.3.1 MLP Input Layer Representations

6.2.1 Ordinal Inputs

An alternative to the ordinal inputs 2 to 4 and 14 to 19 being represented as all zeros and a one for each input pattern, a more relative distribution can be used as an input scheme such as a Gaussian distribution with the peak being the 1 input in the binary scheme. The conversion scheme employed is as follows:

Peak	Distance 1	Distance 2	Distance 3	Distance 4	Distance 5
1	0	0	0	0	0
0.7979	0.108	2.67×10^{-4}	1.21×10^{-8}	1.1×10^{-14}	1.53×10^{-22}

Table 6.3.2 Ordinal Input Conversion Scheme

This conversion is based on the normal probability density function:

$$y = \frac{1}{\sigma\sqrt{2\pi}} \exp[-0.5(\frac{x-\mu}{\sigma})^2] \qquad (6.1)$$

where x is the original binary value, y being the new value and μ being the location of the peak or 1 value from the input. The σ term is set to 0.5

6.2.2 Missing Data

Input patterns with missing data can be included in network training by way of substituting the missing data inputs by an equal non-zero value for each missing input. This value is simply determined as the reciprocal fraction of the number of inputs assigned to that input that is missing.

6.3 MLP Simulation

The performance of Multi Layer Perceptron networks are effected by a wide variety of attributes mainly in two areas, first the MLP architecture, the number of input neurons dictates the dimensionality of the pattern space in which MLP linear decision boundaries are placed. The number of hidden layer neurons dictates the number decision boundaries. The number of outputs dictates the number of possible classifications. The second area is the data used in training. Large sets of training patterns are useful however a high number iterations in training with the same data set can cause *over training.* This occurs when the ANN trains too specifically to the training set and is not generalised enough. This can be measured if a validation set is used, this data sets are composed of patterns not found in the training set but is also used to evaluate the network's performance. The risk of over training is accounted for by weight decay regularisation described in section ().

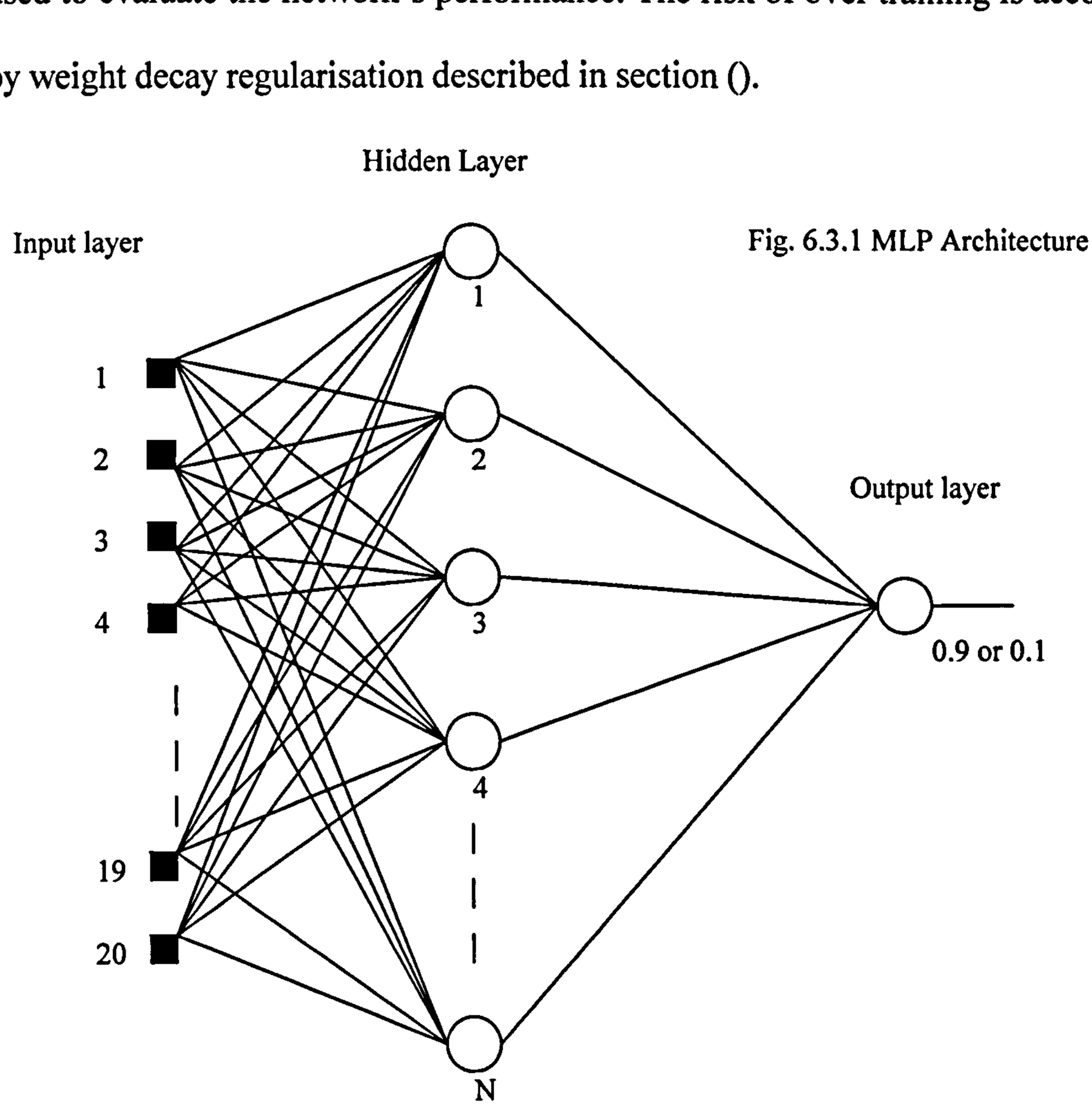
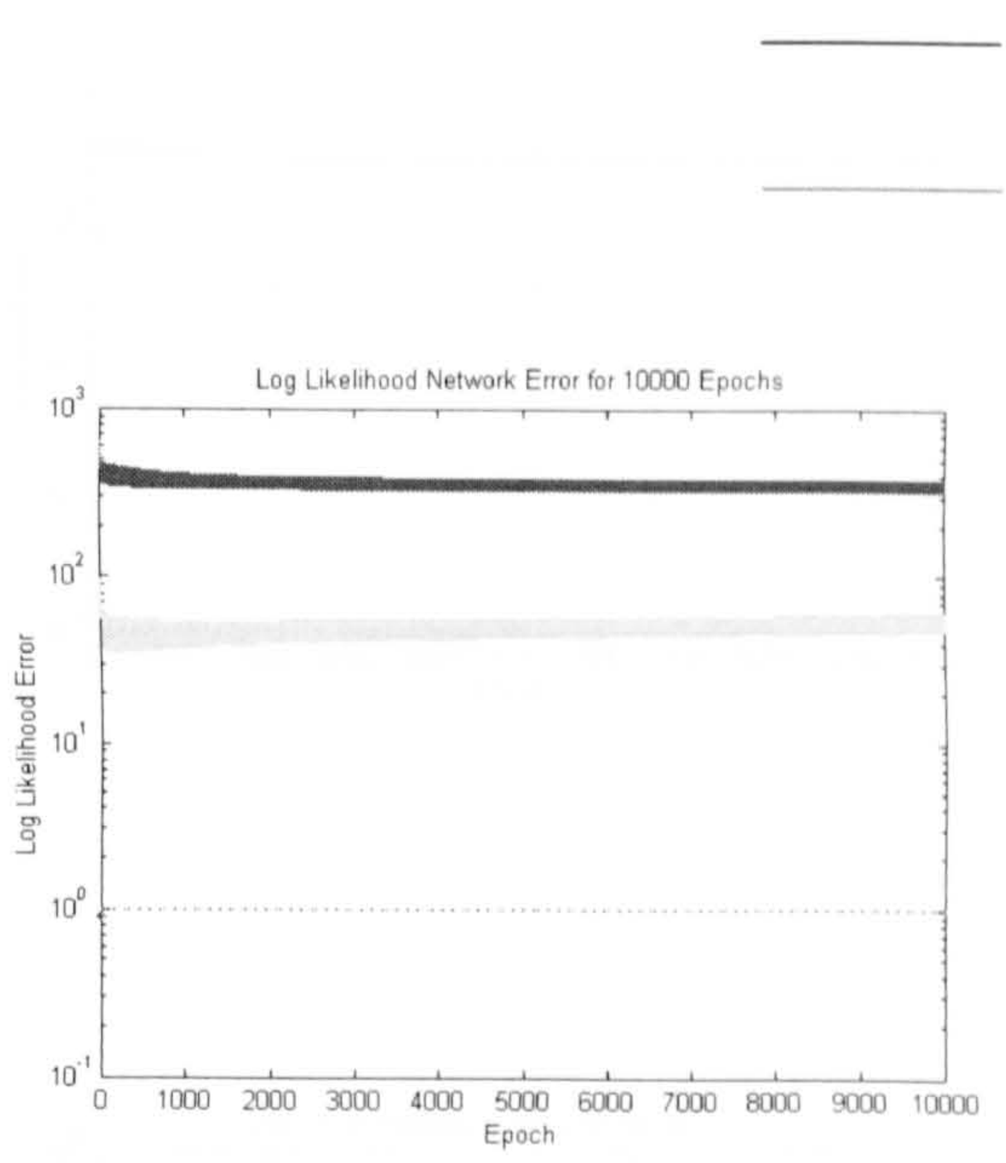


Fig. 6.3.1 MLP Architecture

6.3.1 Twelve Fold Cross Validation Results

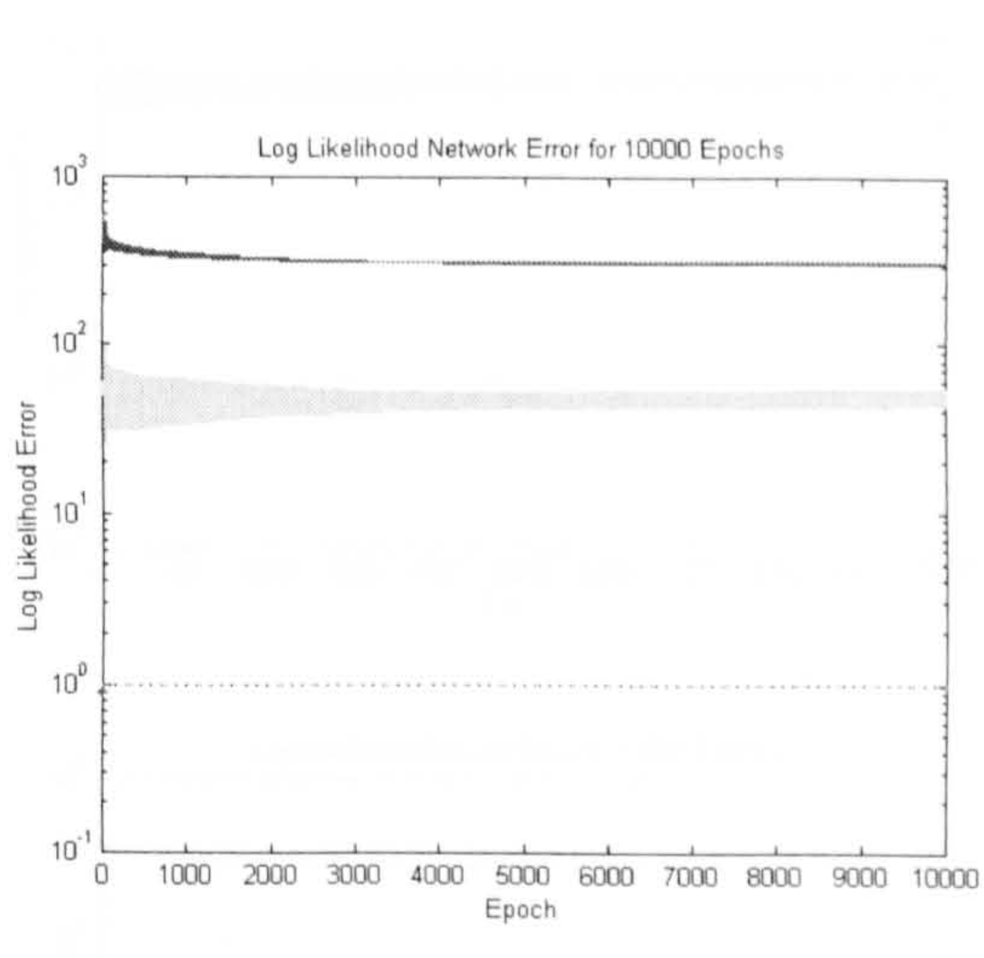
Validation Block 1:47



Validation Block, LLE	Training Block, LLE
48.640	320.054

Fig 6.3.2

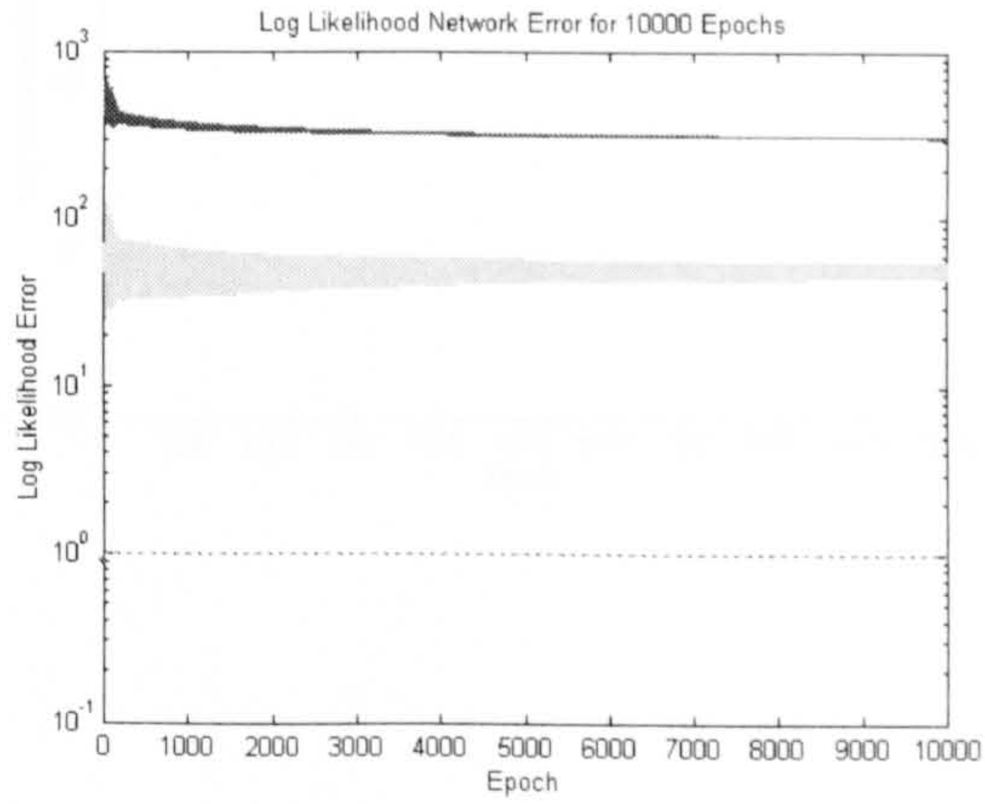
Validation Block 48:94



Validation Block, LLE	Training Block, LLE
43.2477	304.788

Fig 6.3.3

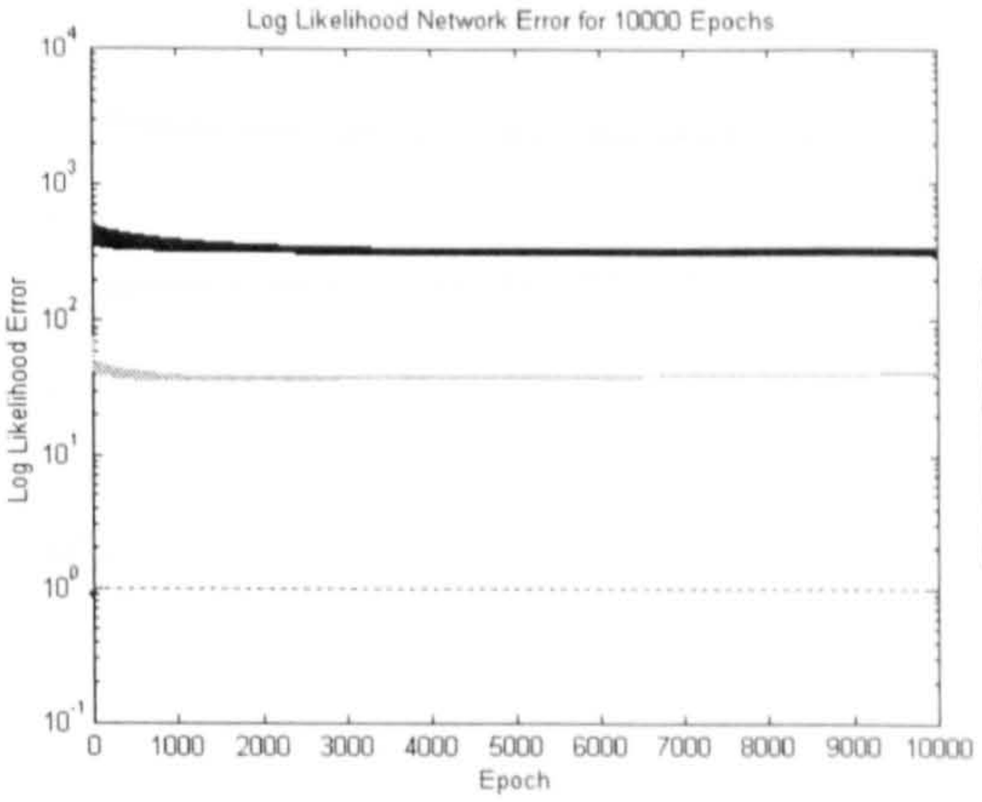
Validation Block 95:141



Validation Block, LLE	Training Block, LLE
44.9531	308.676

Fig 6.3.4

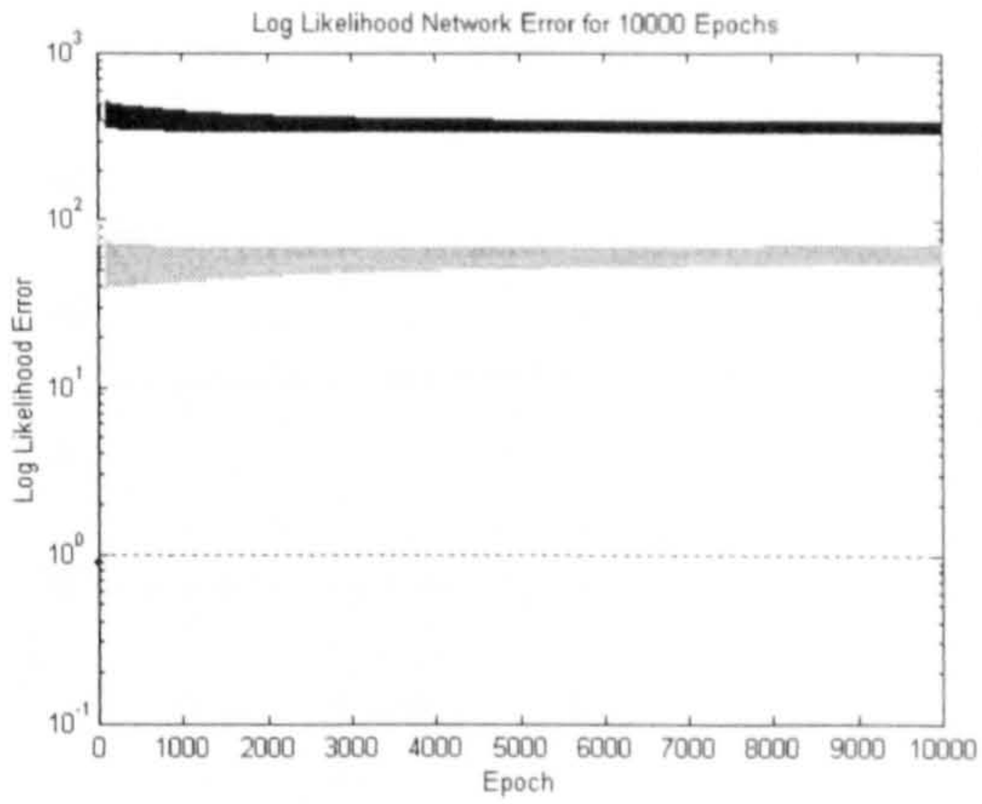
Validation Block
142:188



Validation Block, LLE	Training Block, LLE
40.4177	311.676

Fig 6.3.5

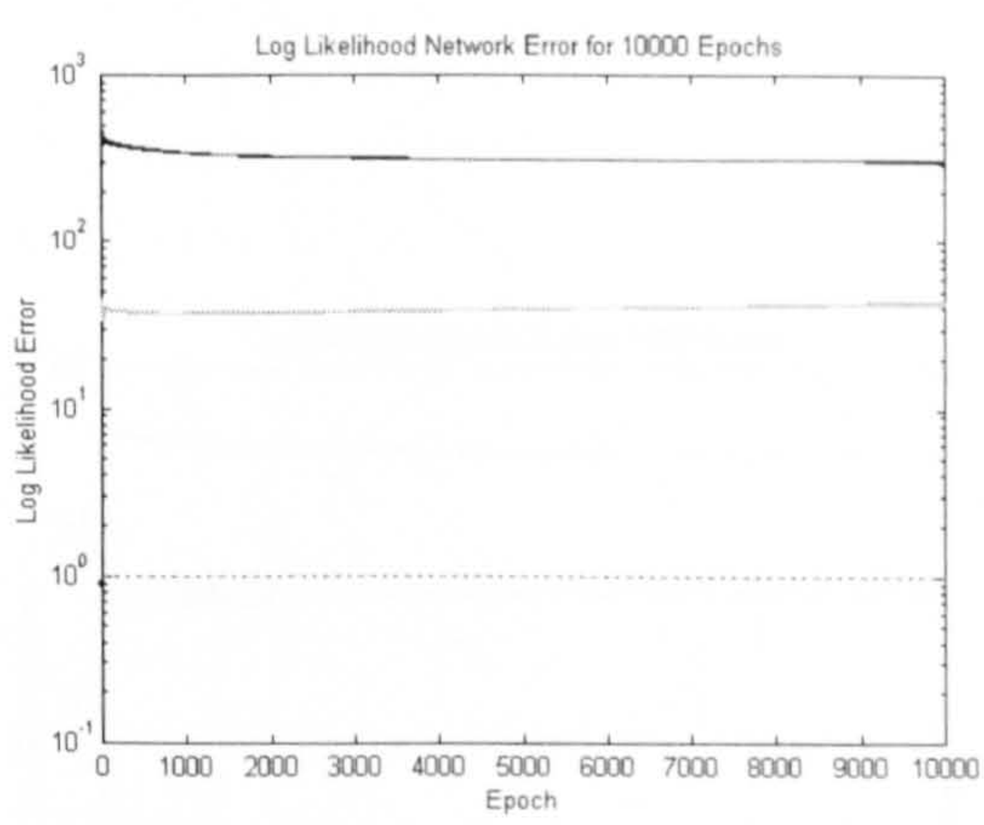
Validation Block
189:235



Validation Block, LLE	Training Block, LLE
55.844	338.94

Fig 6.3.6

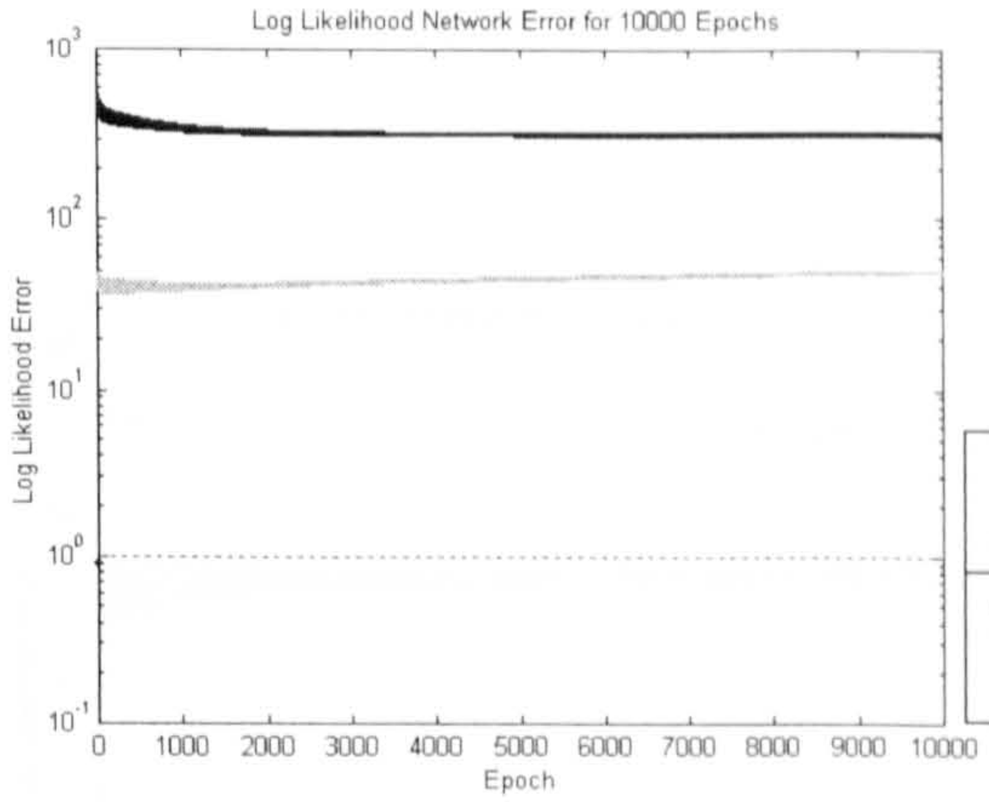
Validation Block
236:282



Validation Block, LLE	Training Block, LLE
43.71	310.198

Fig 6.3.7

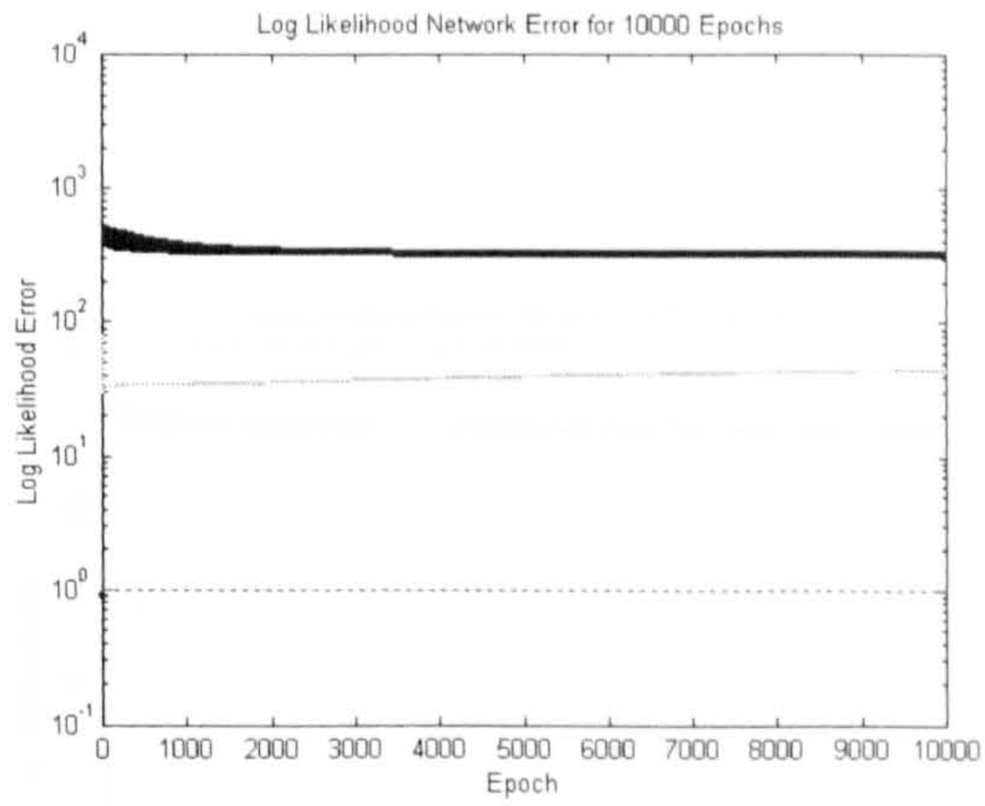
Validation Block
283:329



Validation Block, LLE	Training Block, LLE
48.19	304.482

Fig 6.3.8

Validation Block
330:376



Validation Block, LLE	Training Block, LLE
45.5	310.58

Fig 6.3.9

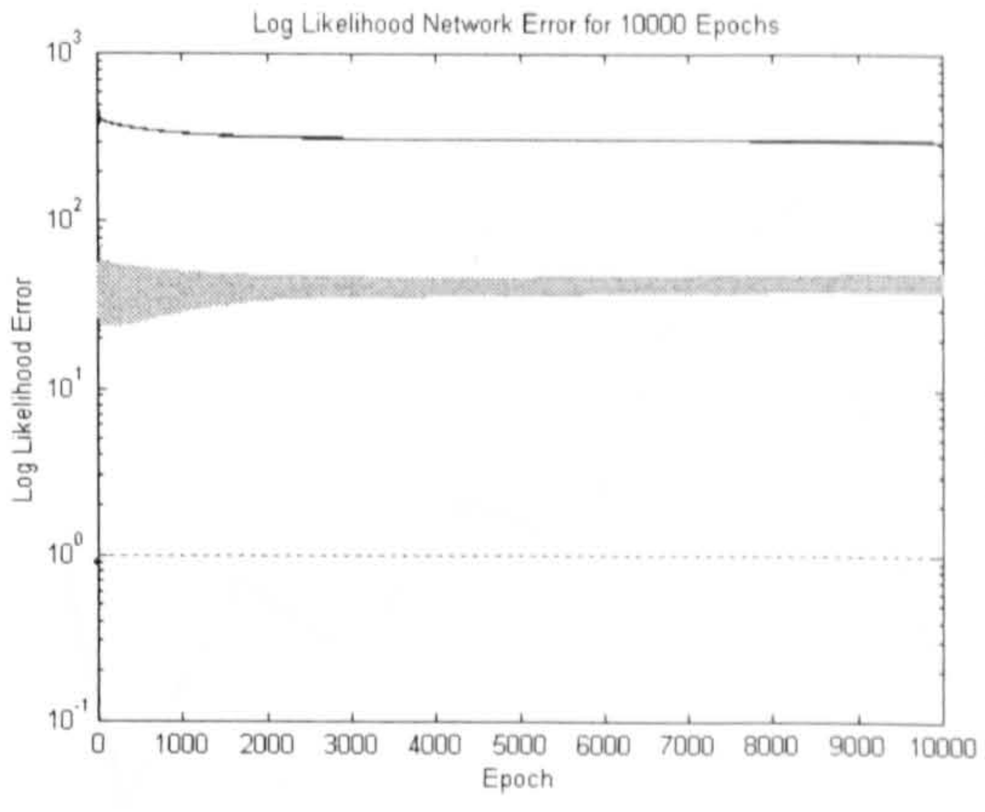
Validation Block
377:423

Validation Block, LLE	Training Block, LLE
38.3	306.945

Fig 6.3.10

Validation Block
424:470

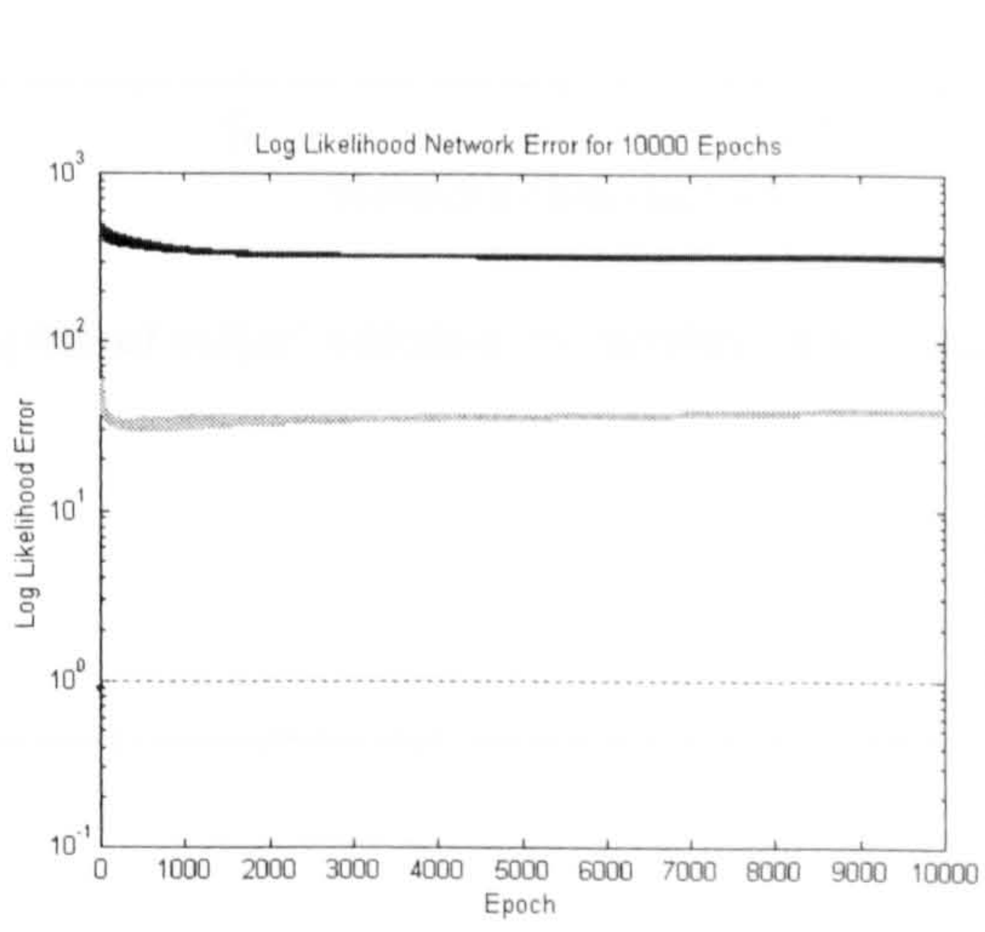
Fig 6.3.11



Validation Block, LLE	Training Block, LLE
43.7144	312.113

Validation Block
471:517

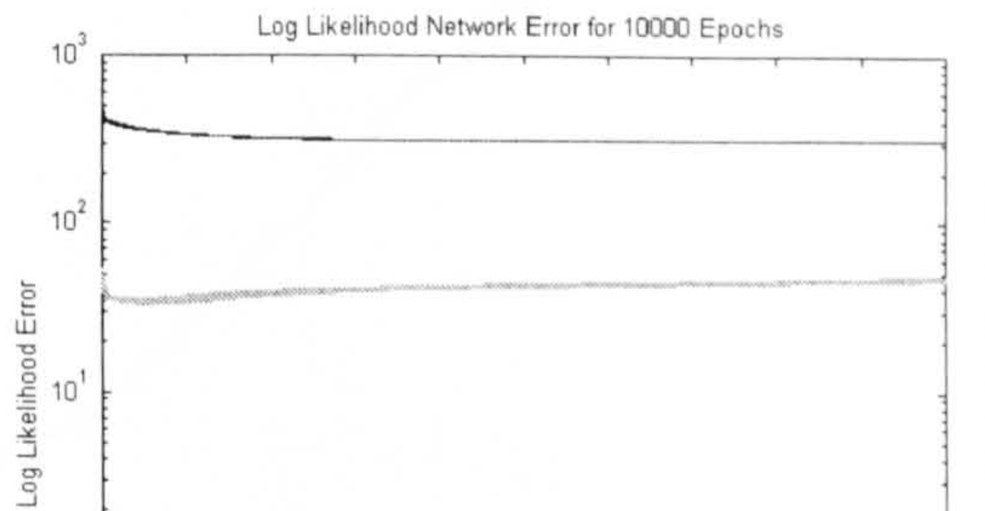
Fig 6.3.12



Validation Block, LLE	Training Block, LLE
28.72	335.692

Validation Block
518:564

Fig 6.3.13



Validation Block, LLE	Training Block, LLE
312.129	45.6

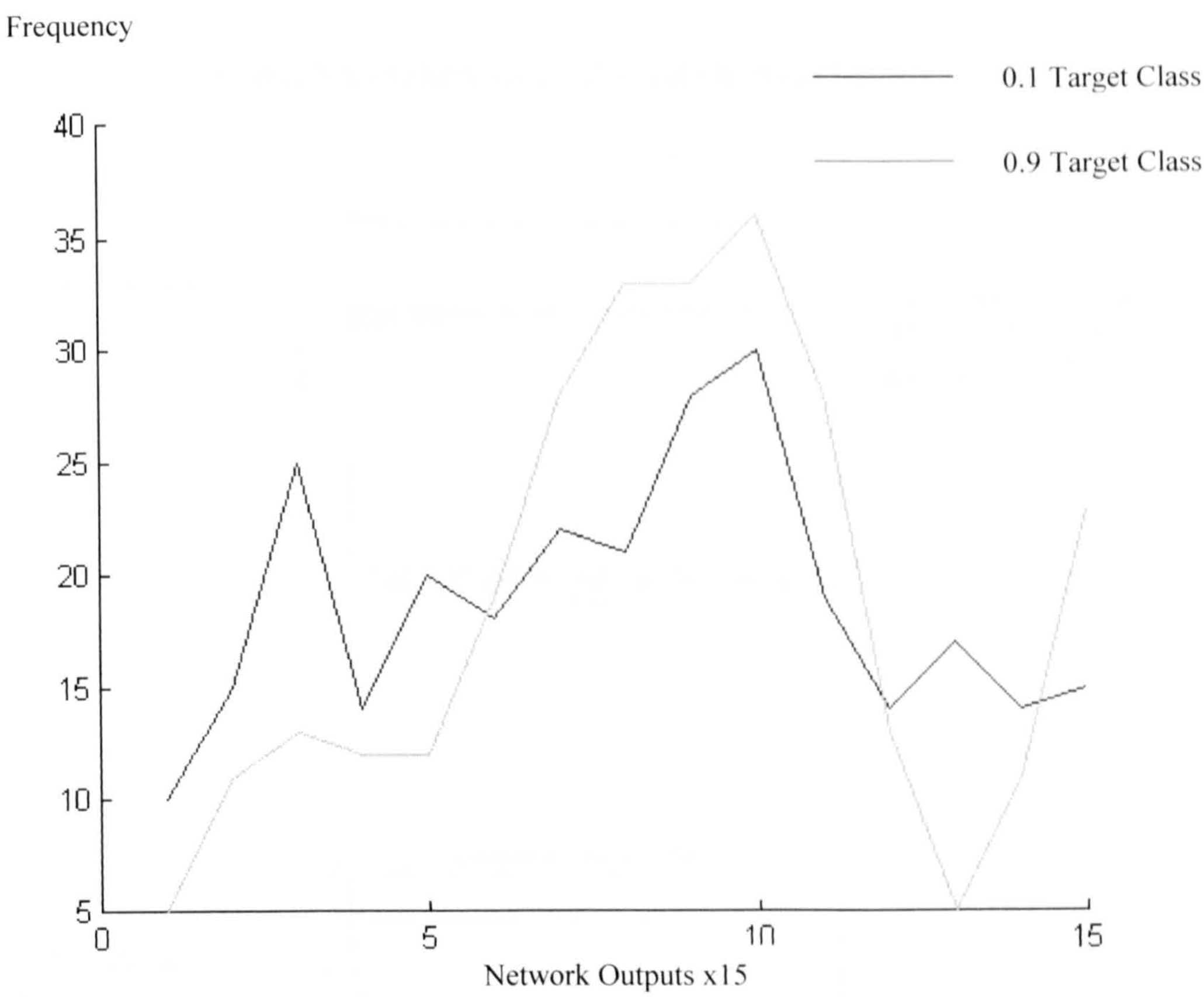


Figure 6.3.14: Histogram of output Values x 15 (totalled in 15 bins) for regular 12 fold cross validation

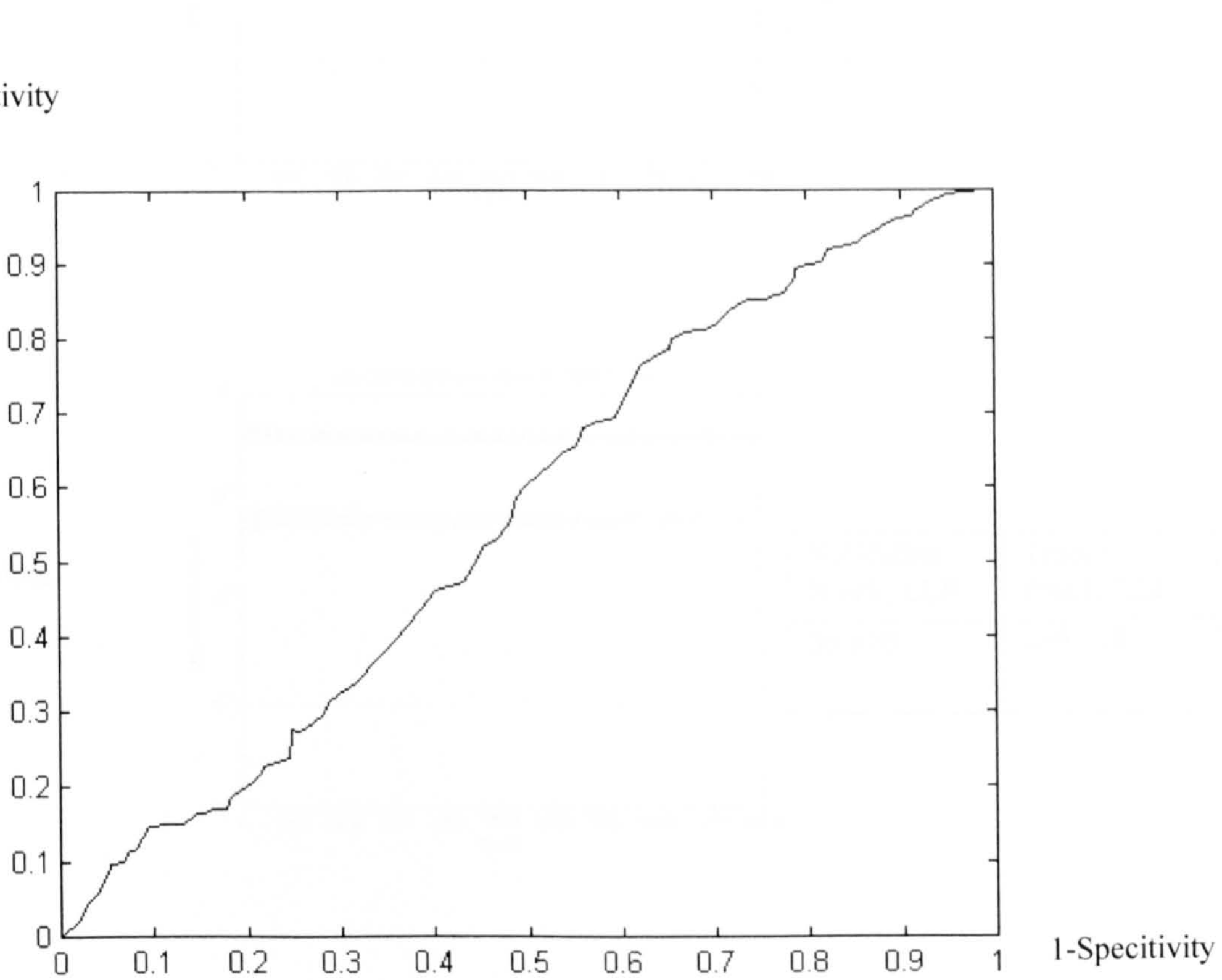


Fig 6.3.15: Receiver Operator Characteristic for 12 fold cross validation

6.3.2 Twelve Fold Cross Validation with Altered Ordinal Inputs Results

Validation Block 1:47

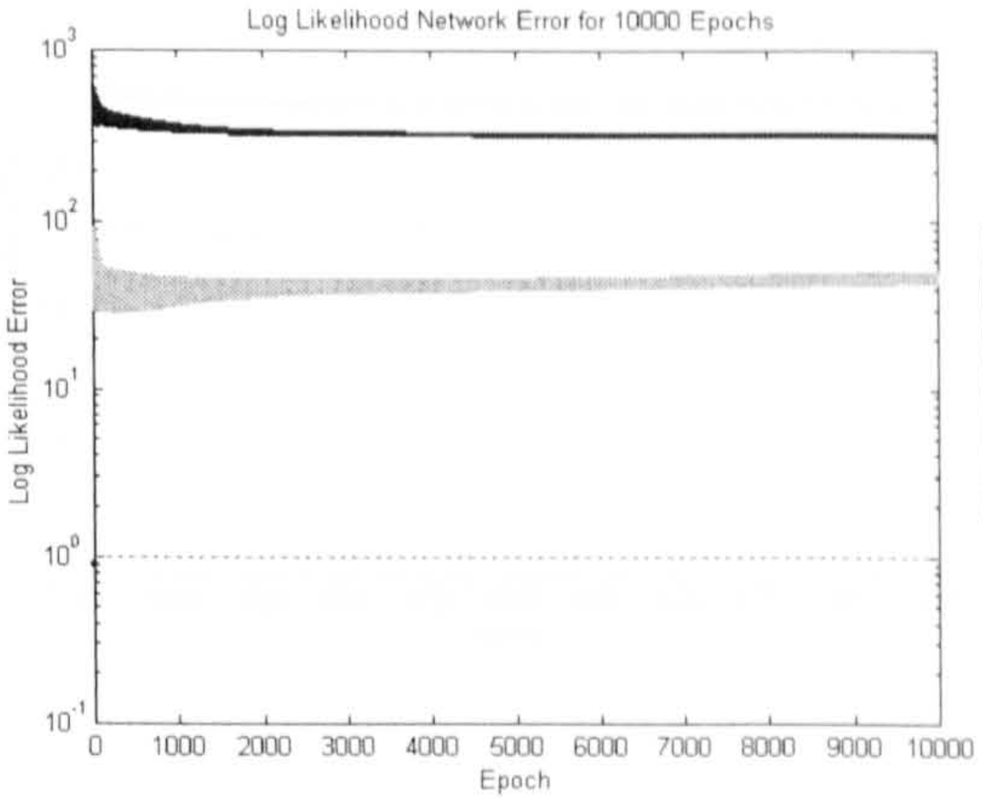


Fig 6.3.16

Validation Block, LLE	Training Block, LLE
43.598	313.164

Validation Block 48:94

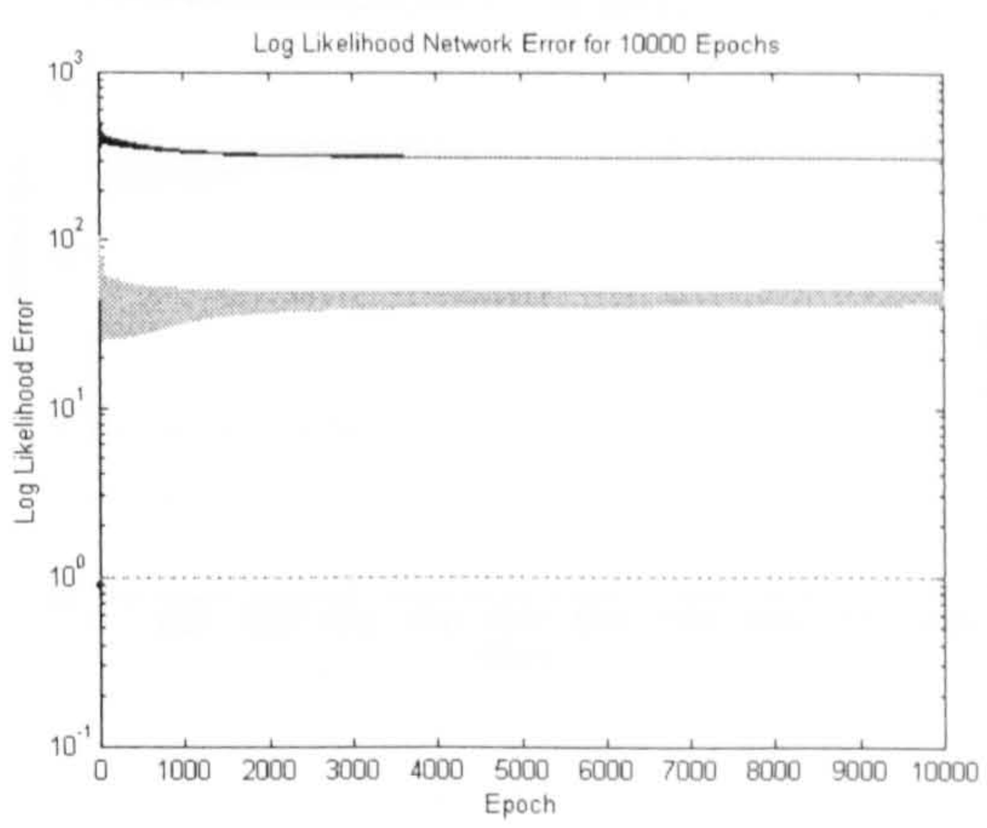


Fig 6.3.17

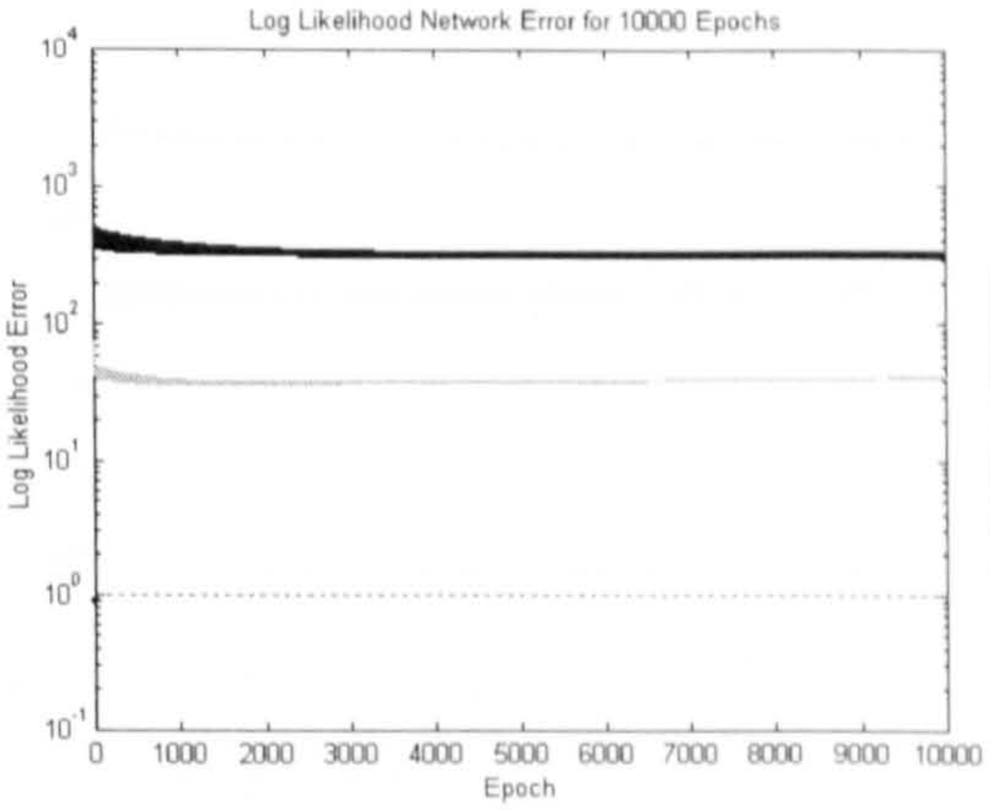
Validation Block, LLE	Training Block, LLE
42.210	309.146

Validation Block 95:141

Fig 6.3.18

Validation Block, LLE	Training Block, LLE
55.606	350.128

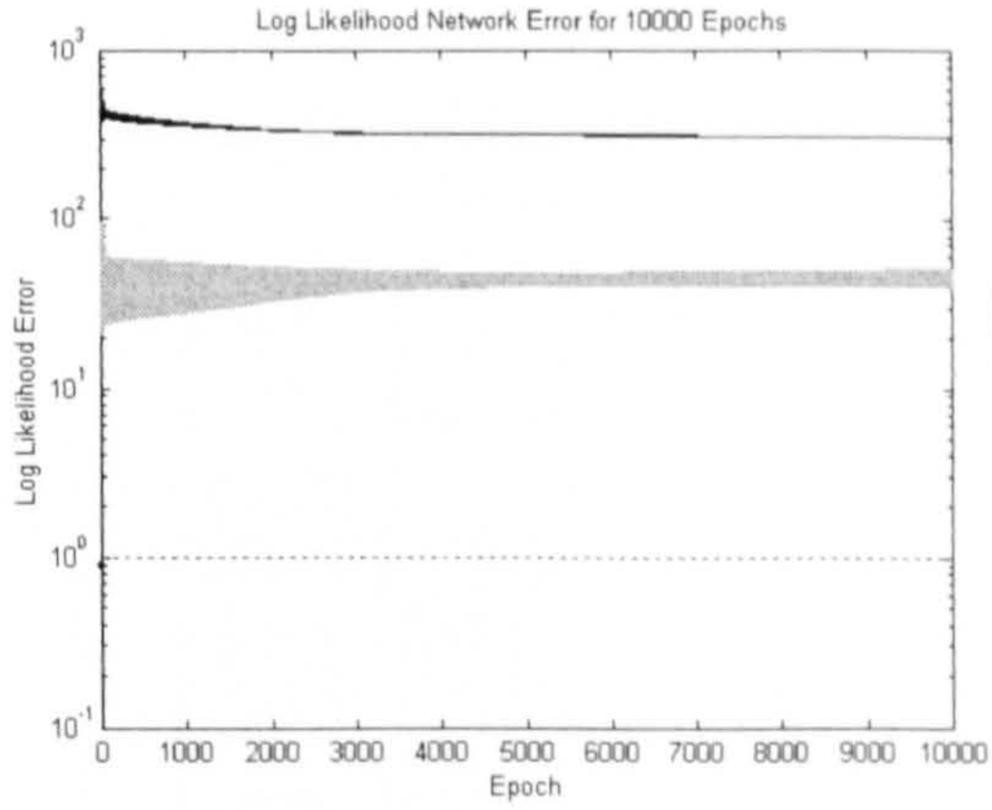
Validation Block
142:188



Validation Block, LLE	Training Block, LLE
40.4177	311.676

Fig 6.3.19

Validation Block
189:235



Validation Block, LLE	Training Block, LLE
40.55	314.785

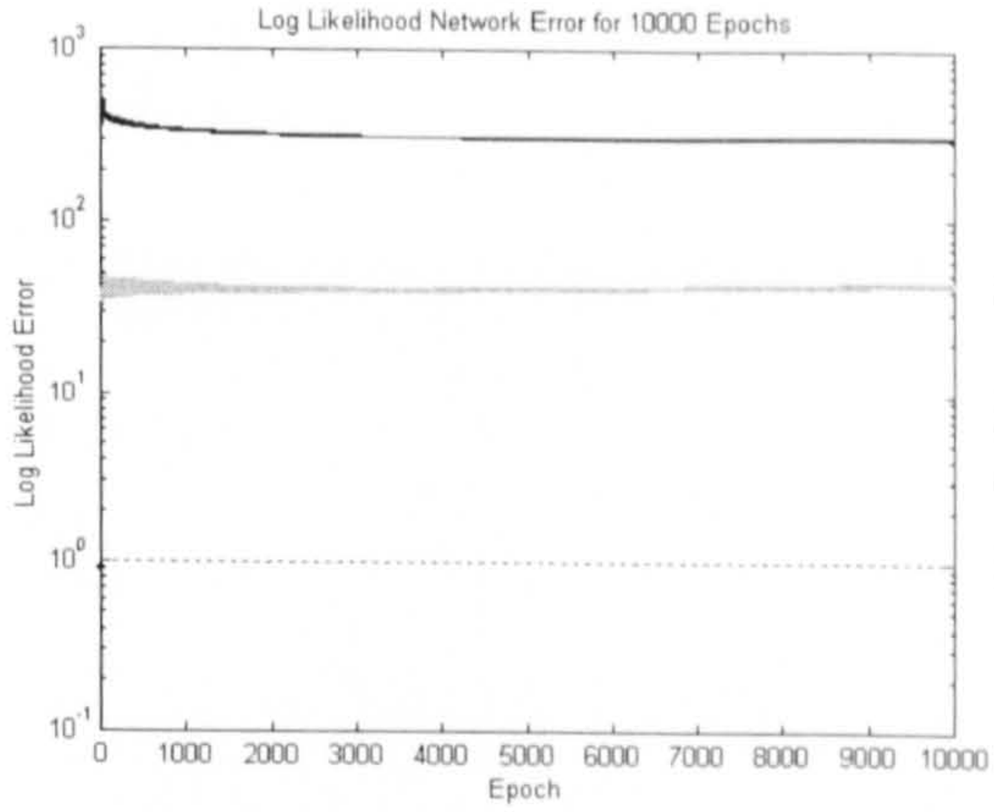
Fig 6.3.20

Validation Block
236:282

Validation Block, LLE	Training Block, LLE
43.27	311.542

Fig 6.3.21

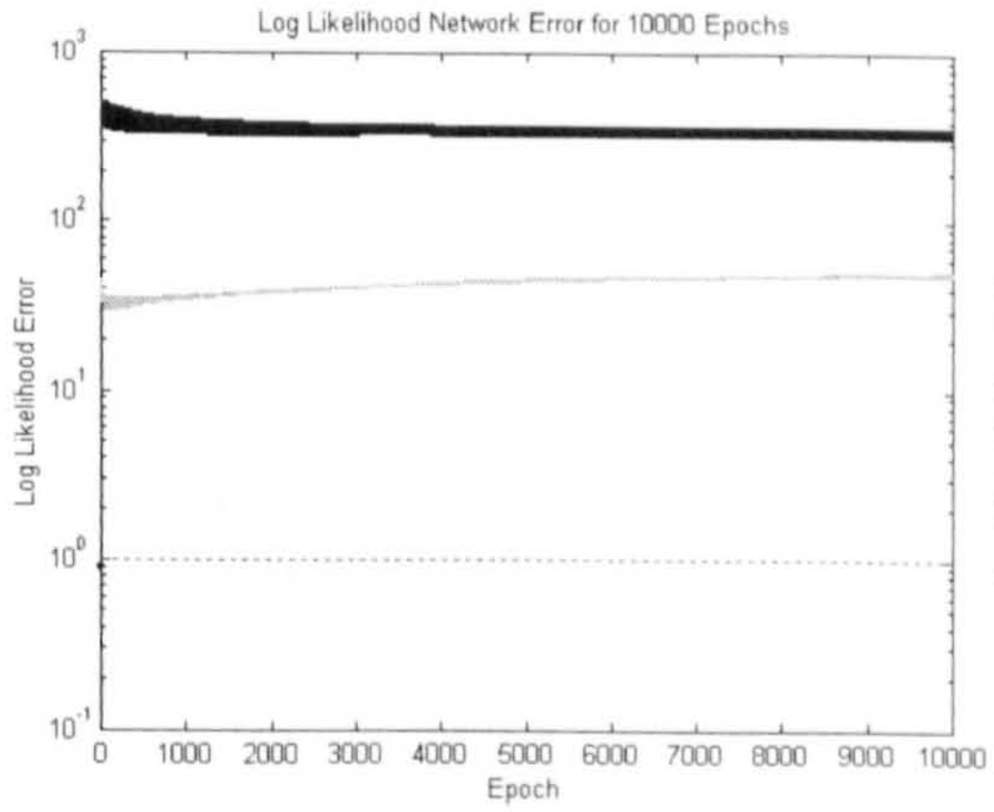
Validation Block
283:329



Validation Block, LLE	Training Block, LLE
45.159	304.573

Fig 6.3.22

Validation Block
330:376



Validation Block, LLE	Training Block, LLE
49.165	319.303

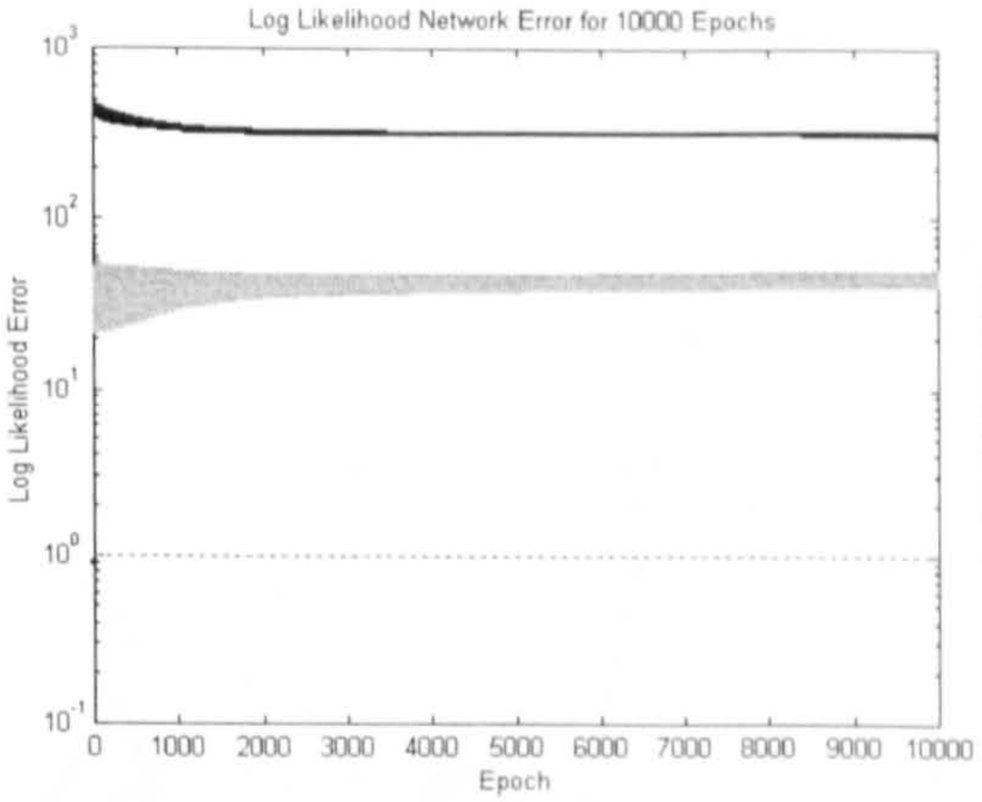
Fig 6.3.23

Validation Block
377:423

Validation Block, LLE	Training Block, LLE
45.6298	352.715

Fig 6.3.24

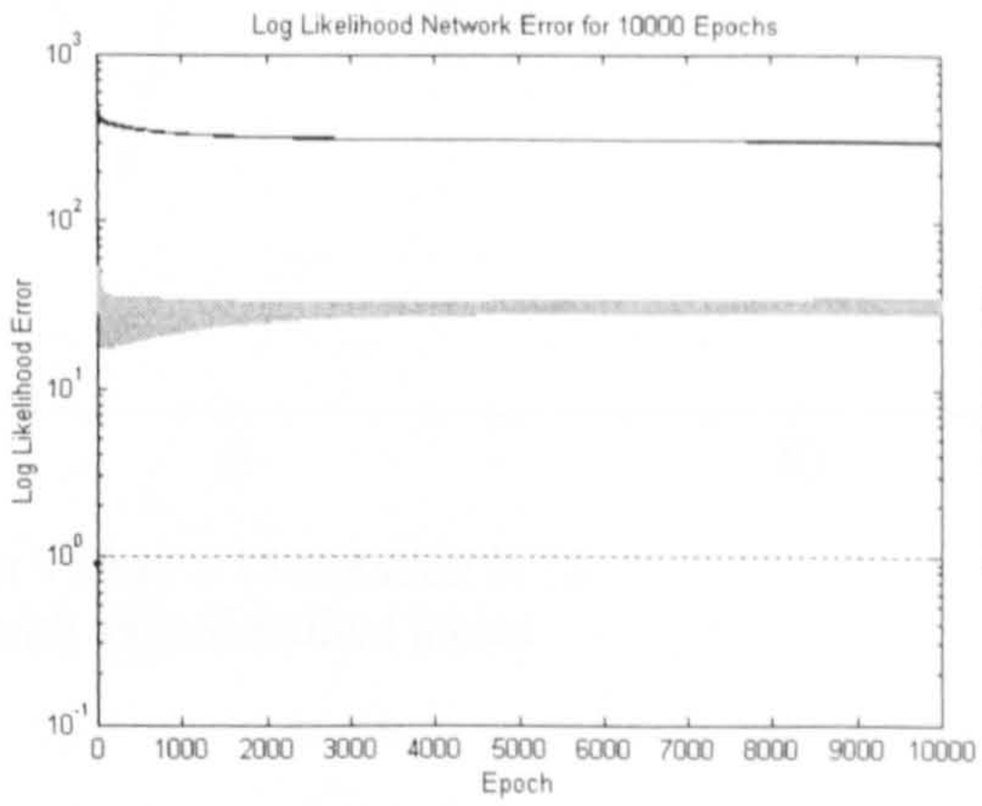
Validation Block
424:470



Validation Block, LLE	Training Block, LLE
50.6164	308.022

Fig 6.3.25

Validation Block
471:517



Validation Block, LLE	Training Block, LLE
28.6766	309.744

Fig 6.3.26

Validation Block
518:564

Validation Block, LLE	Training Block, LLE
40.1152	341.581

Fig 6.3.27

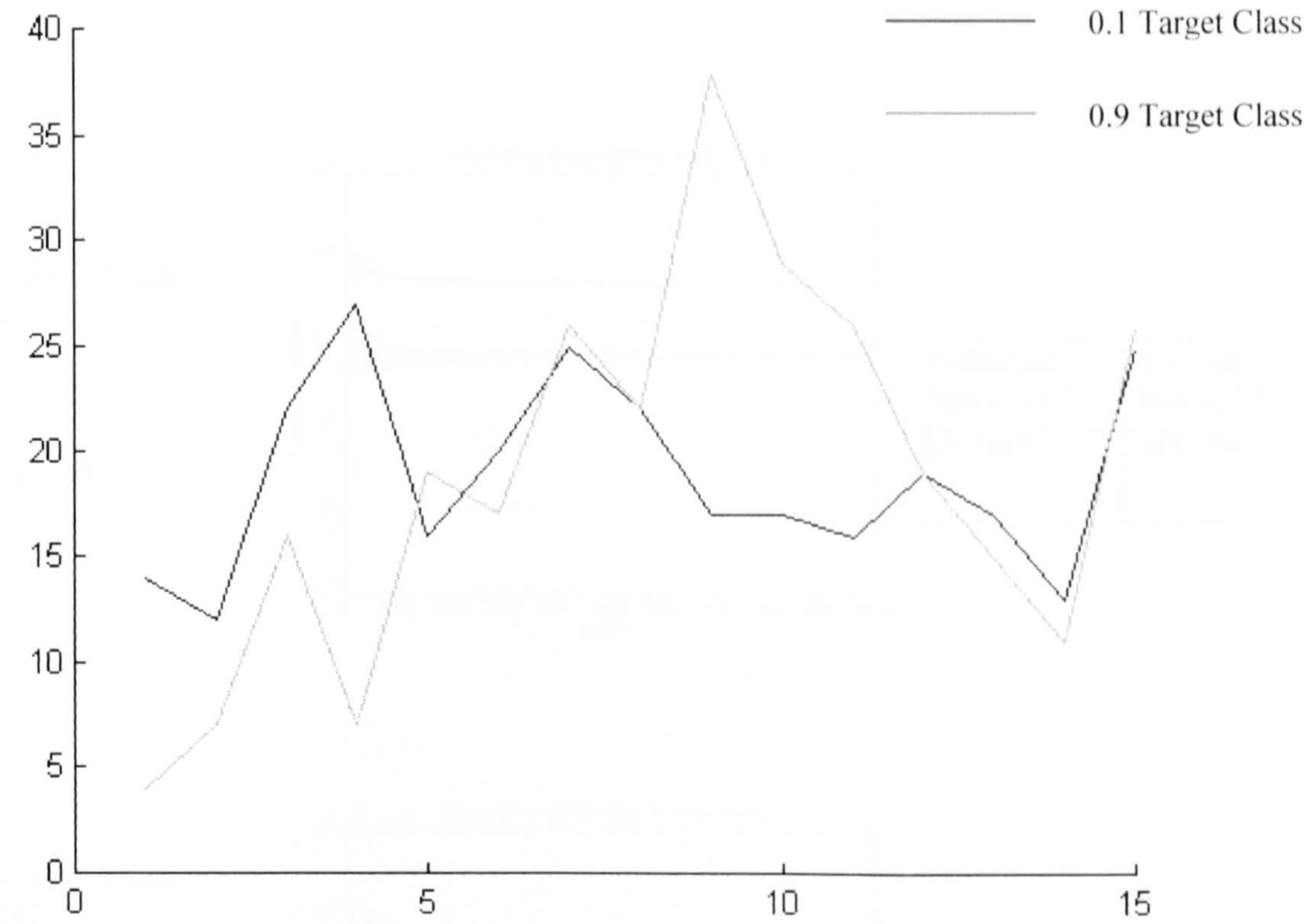


Figure 6.3.28: Histogram of output Values x 15 (totalled in 15 bins) for 12 fold cross validation with altered ordinal inputs

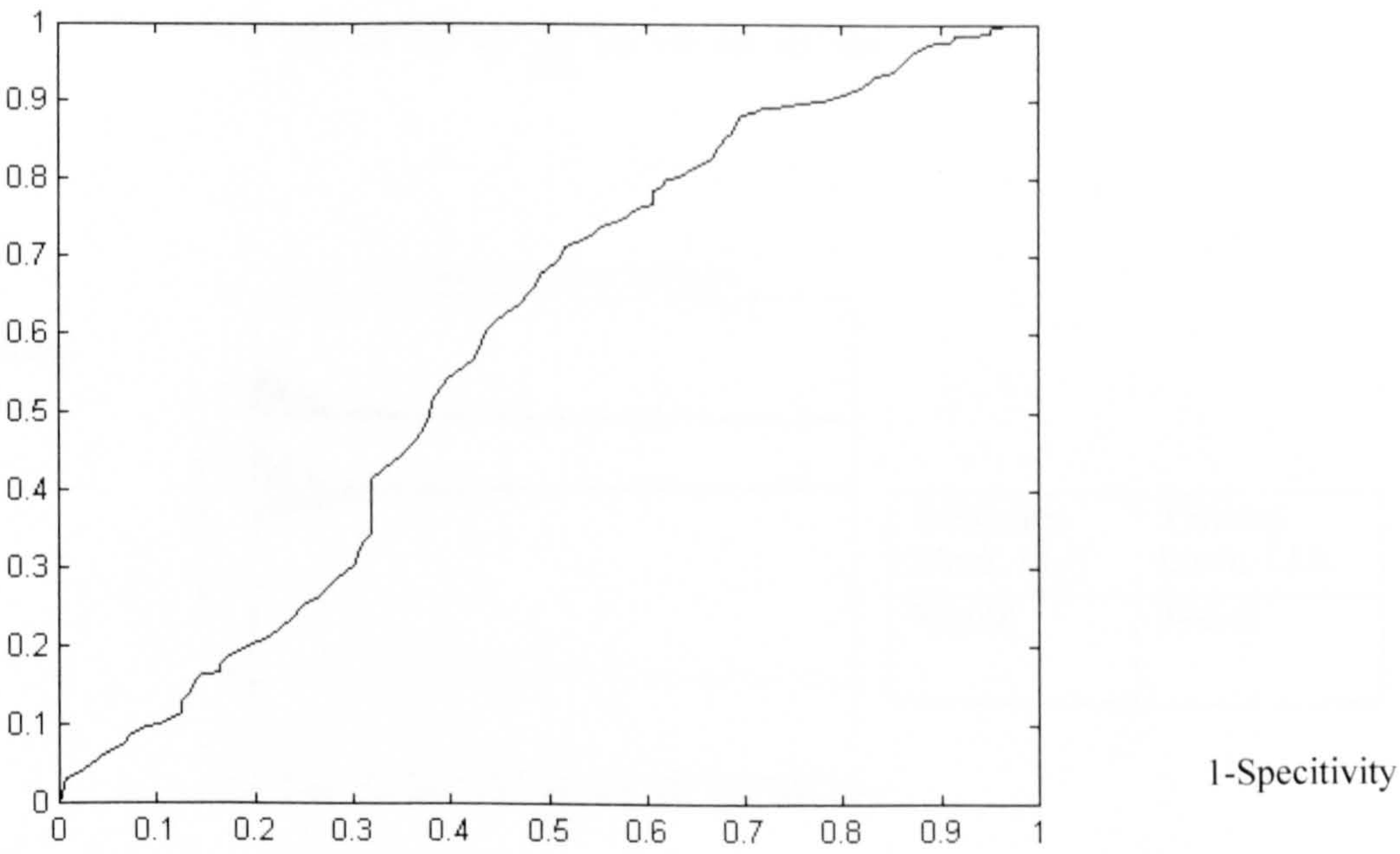


Fig 6.3.29: Receiver Operator Characteristic for 12 fold cross validation with altered ordinal inputs

6.3.3 Twelve Fold Cross Validation with Missing Data

Validation Block 1:71

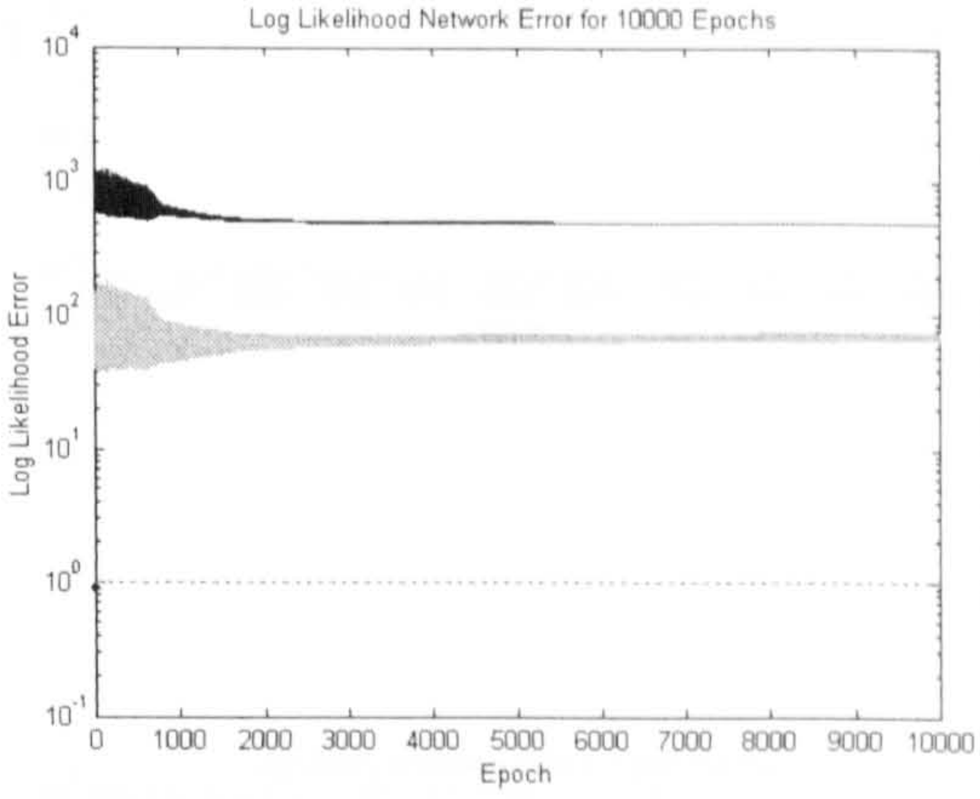


Fig 6.3.30

Validation Block, LLE	Training Block, LLE
67.7547	488.259

Validation Block 72:142

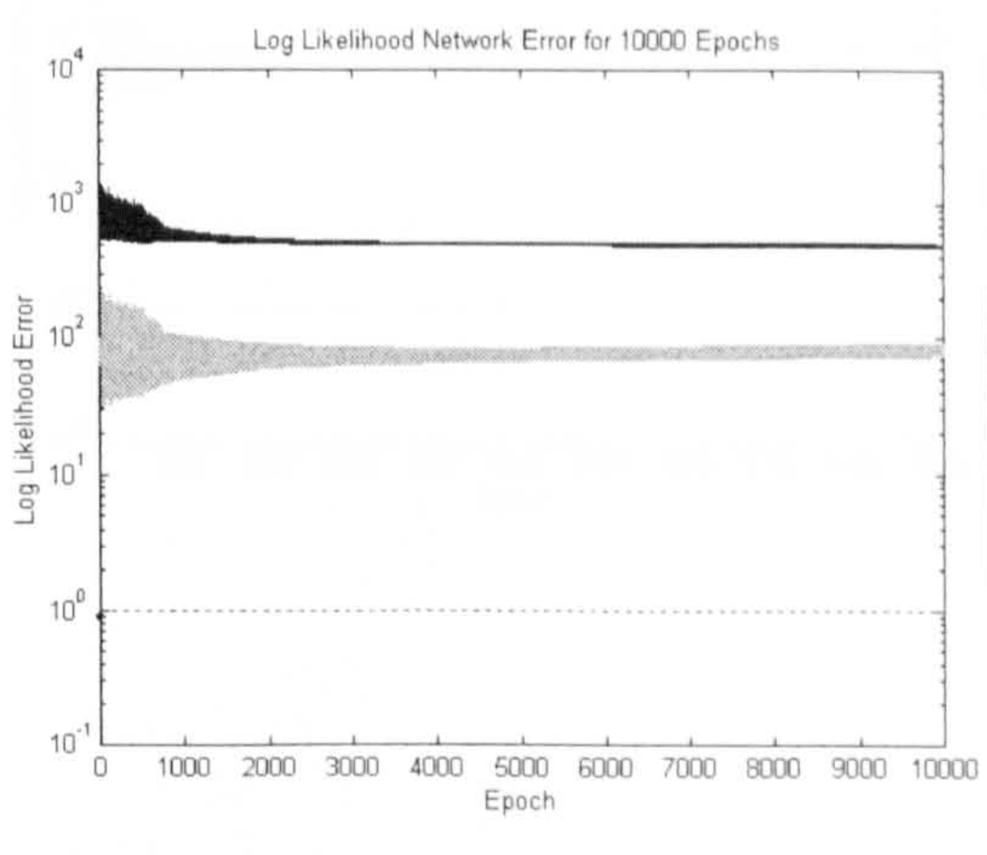


Fig 6.3.31

Validation Block, LLE	Training Block, LLE
73.758	483.207

Validation Block 143:213

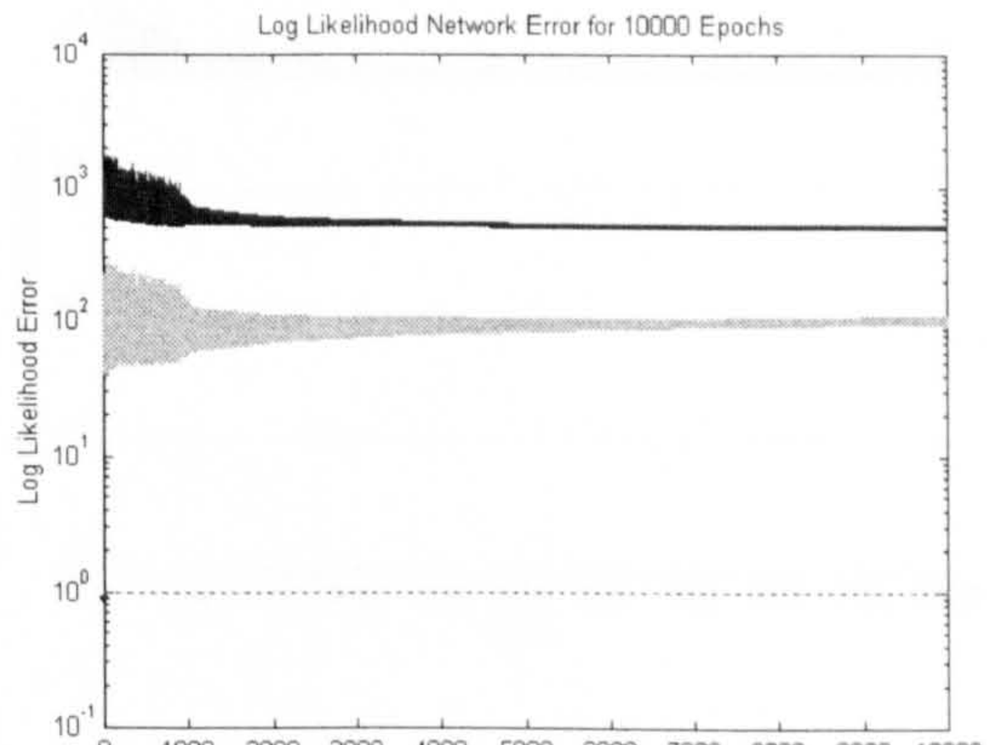
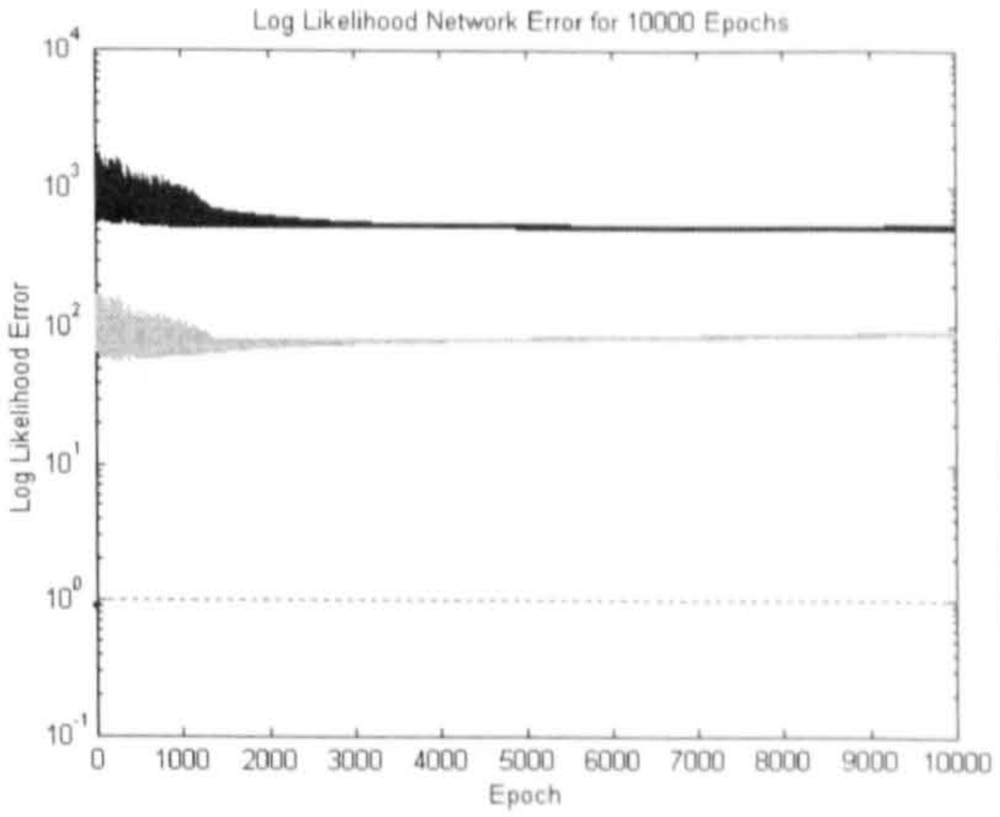


Fig 6.3.32

Validation Block, LLE	Training Block, LLE
95.628	499.46

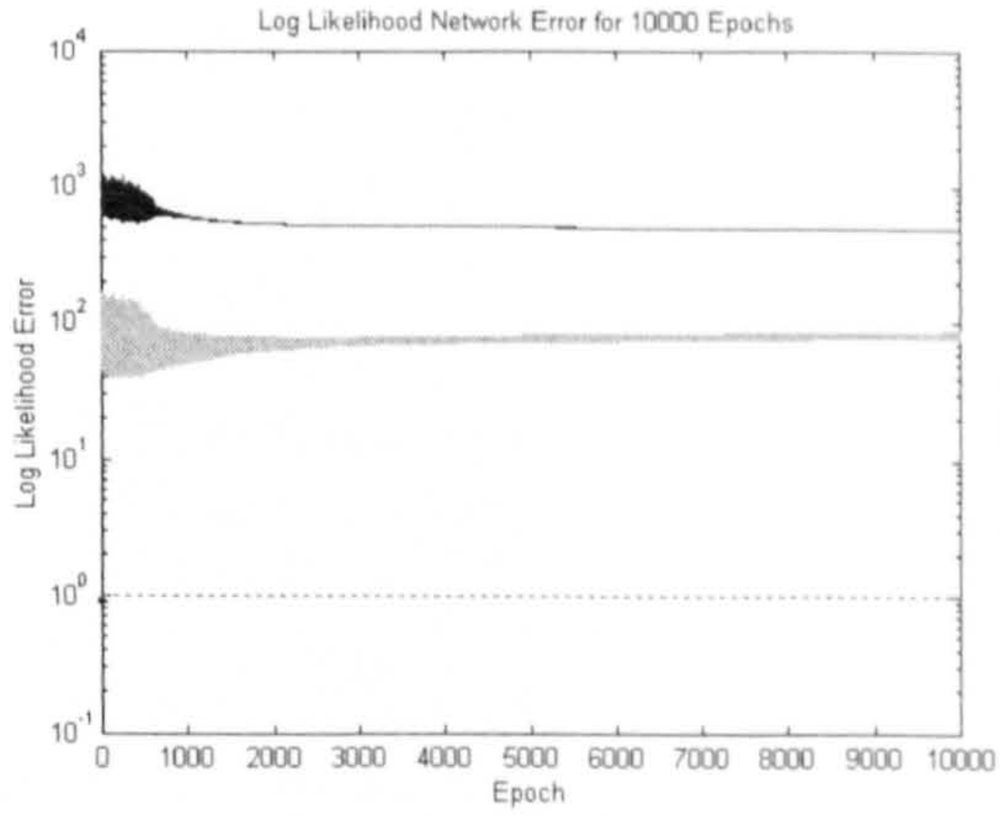
Validation Block
214:284



Validation Block, LLE	Training Block, LLE
88.614	502.86

Fig 6.3.33

Validation Block
285:355



Validation Block, LLE	Training Block, LLE
78.3194	499.058

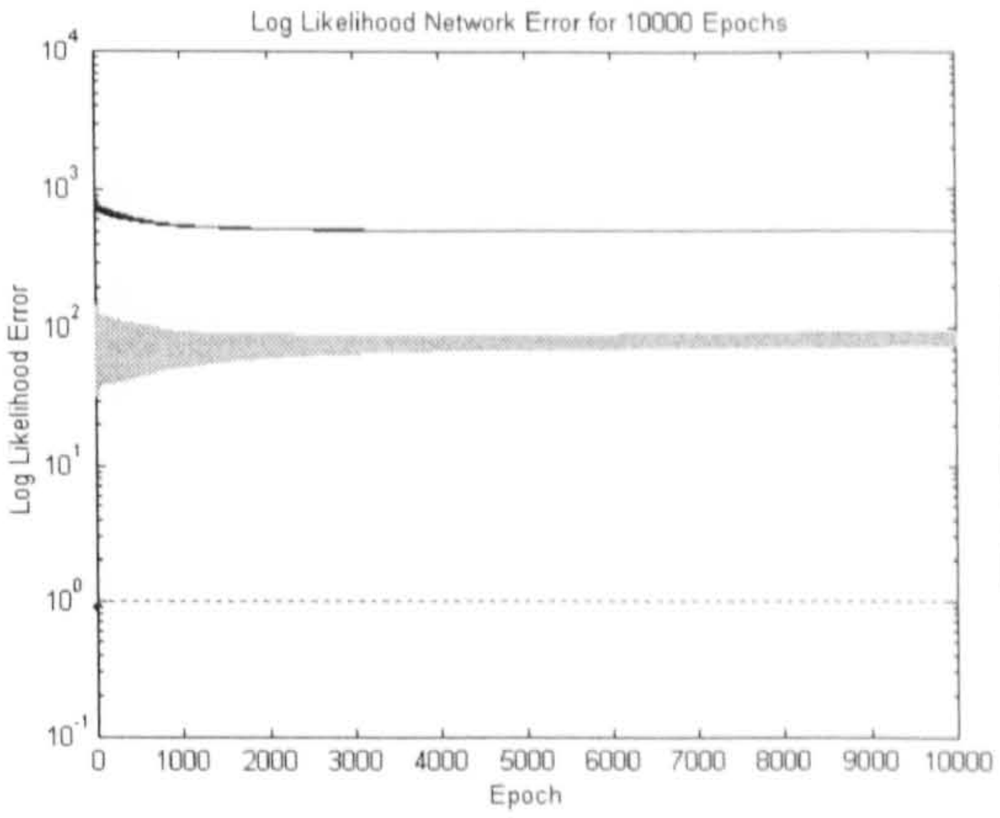
Fig 6.3.34

Validation Block
356:426

Validation Block, LLE	Training Block, LLE
83.117	506.948

Fig 6.3.35

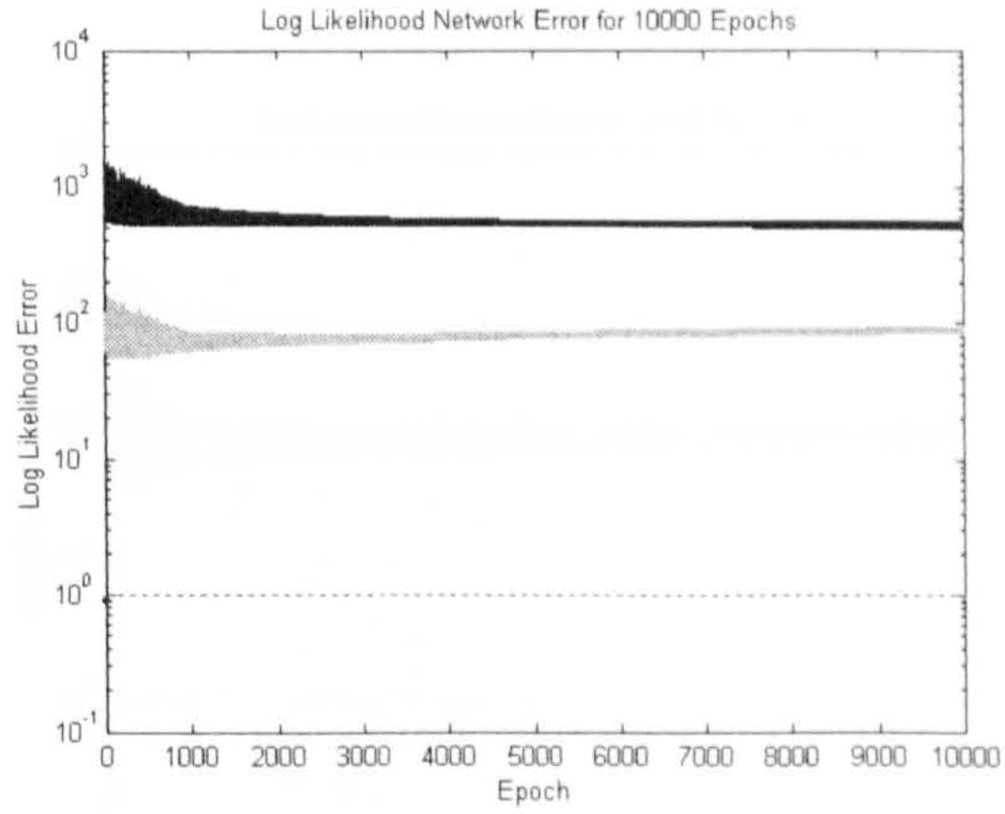
Validation Block
427:497



Validation Block, LLE	Training Block, LLE
96.024	501.803

Fig 6.3.36

Validation Block
498:568



Validation Block, LLE	Training Block, LLE
76.112	519.814

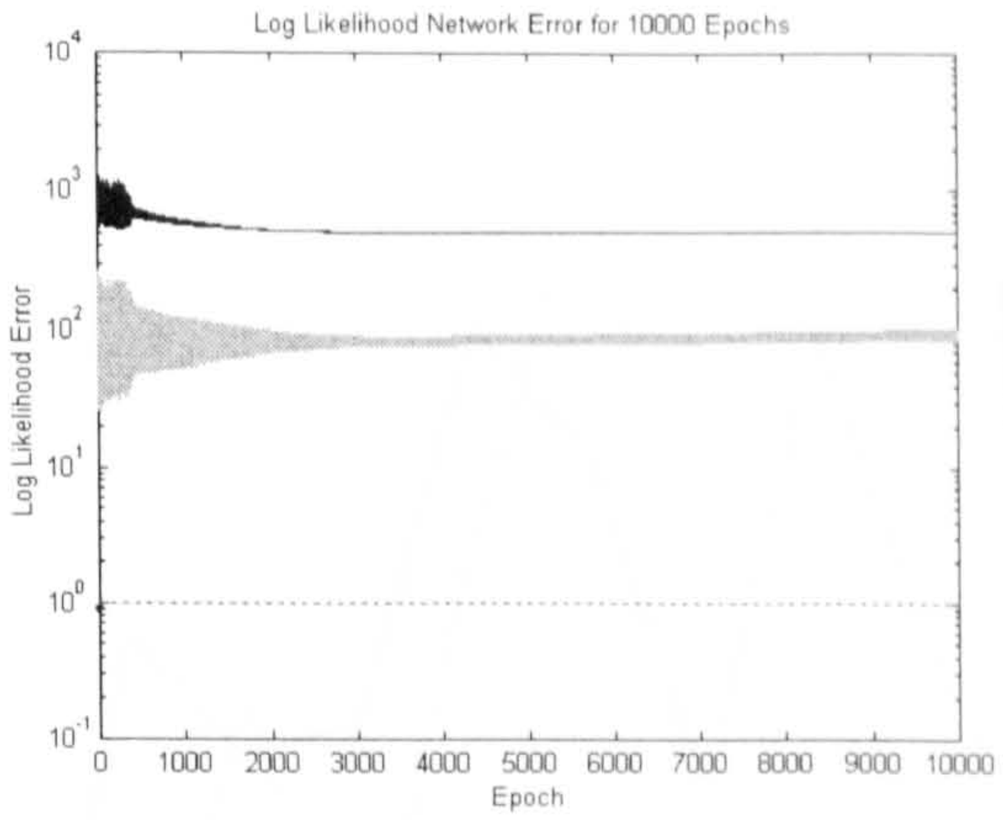
Fig 6.3.37

Validation Block
569:639

Validation Block, LLE	Training Block, LLE
72.9921	485.145

Fig 6.3.38

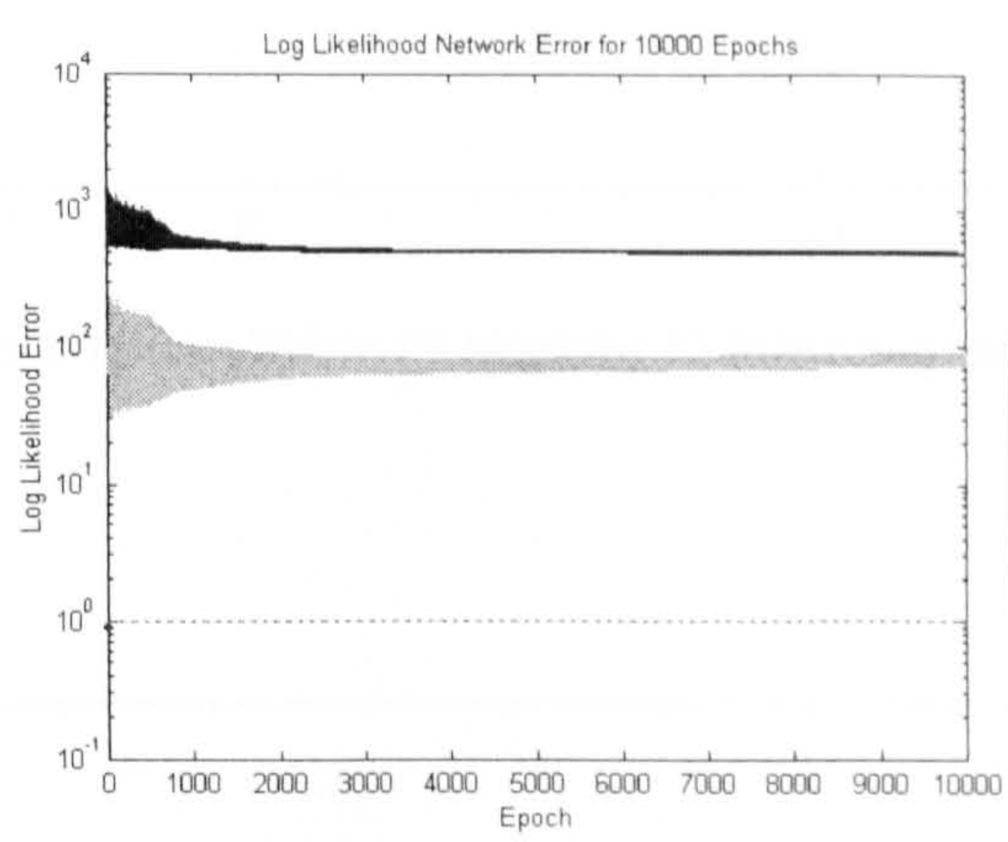
Validation Block
640:710



Validation Block, LLE	Training Block, LLE
85.7468	492.82

Fig 6.3.39

Validation Block
711:781



Validation Block, LLE	Training Block, LLE
90.4012	512.658

Fig 6.3.40

Validation Block
782:852

Validation Block, LLE	Training Block, LLE
81.0707	490.341

Fig 6.3.41

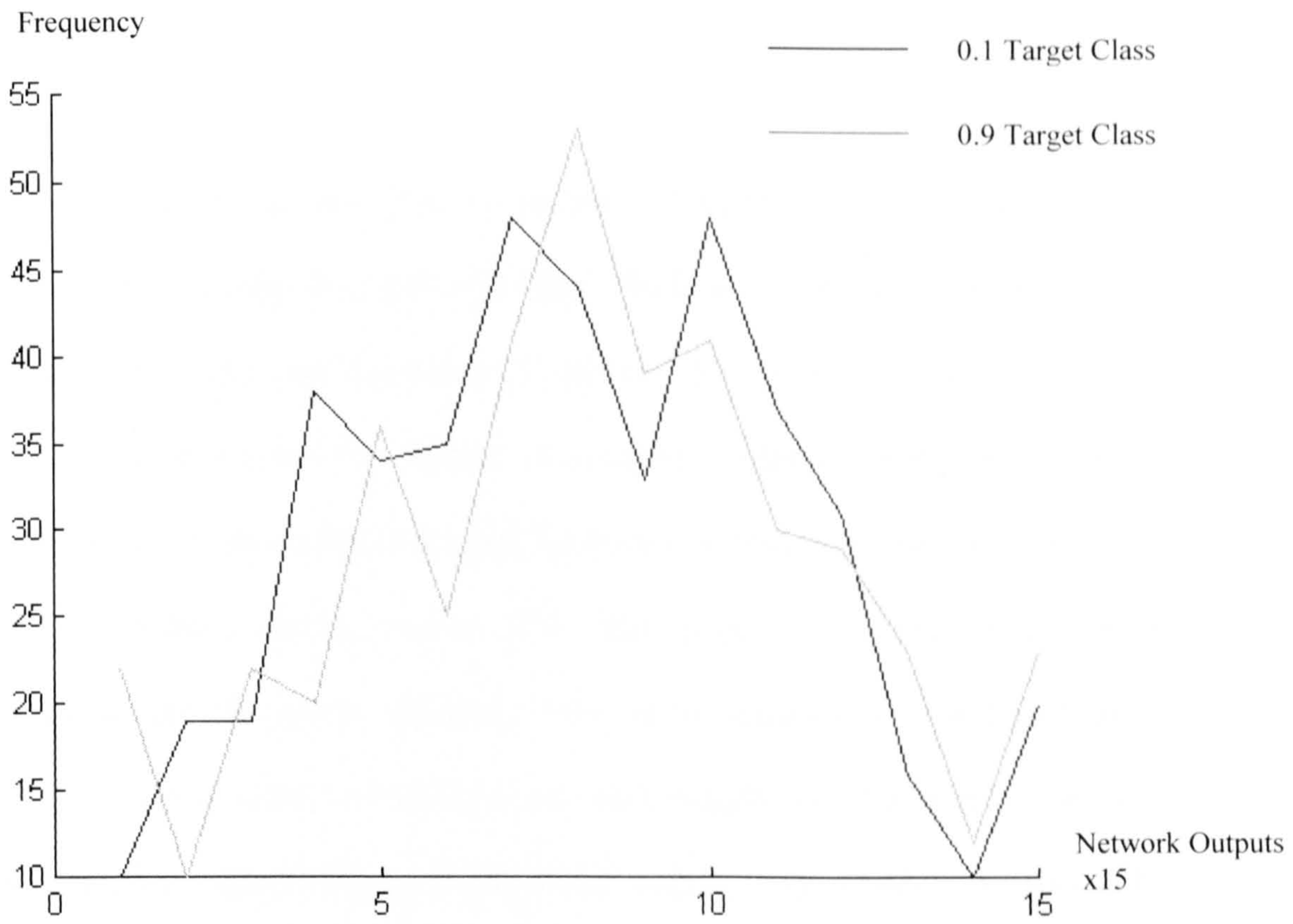


Fig 6.3.42 Histogram of output Values x 15 (totalled in 15 bins) for 12 fold cross validation with missing data inclusive

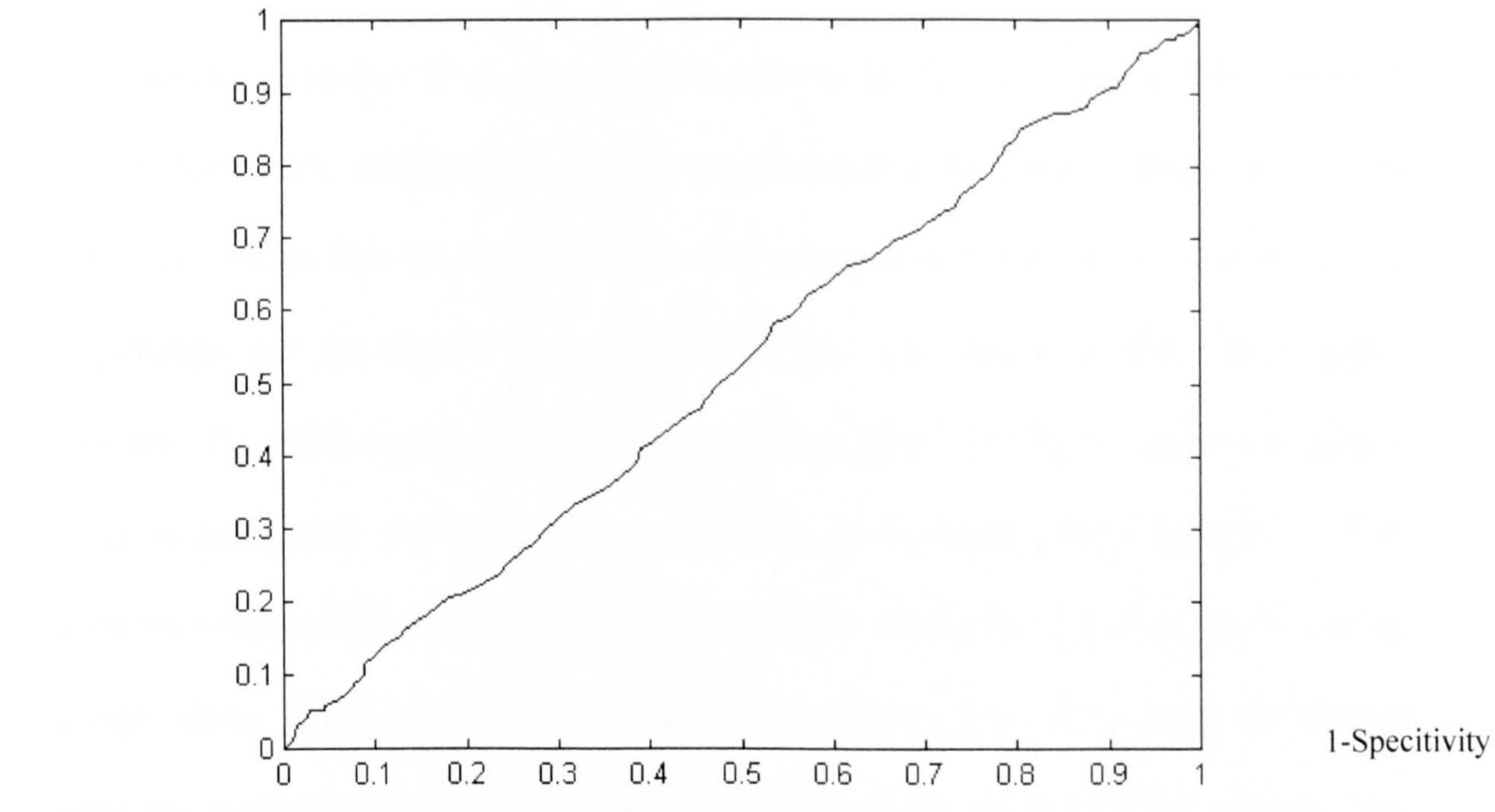


Fig 6.3.43: Receiver Operator Characteristic for 12 fold cross validation with missing data inclusive

6.4 CSF Simulation

Training CSF networks as described in section 3.5 requires the implementation of network initialisation using the OLS algorithm which places the centre points for the hyper-sphere boundaries and consequently dictates the number of hidden neurons. After the OLS initialisation the update of network weights, centres and opening angles is required. However the order and frequency in which the weight, centre and angle sets are updated can be various. CSF training can be considered with both centre updating and non-centre updating. Non centre updating is considered since OLS calculates centre values to minimise the error margin and alteration of the OLS calculation may be a disadvantage. Furthermore opening angles need not necessarily be updated every epoch and can be set to update per set number of epochs between the weight updating.

The number of centres to be placed to initialise the hidden layer size is predetermined. The hidden layer weights are set to zero eliminating the MLP characteristic of the CSF network so that the OLS algorithm can commence. Moreover the opening angle parameters are set to $\pi/4$. This defines the CSF boundaries as RBF style hyper-spheres. The OLS algorithm then places one boundary and the output of the hidden layer is computed. The next stage is to update the output layer's weights. A CSF network output is gained and Log Likelihood Error is calculated and regularised using weight decay. Then another boundary is places using OLS and the process is repeated until the predetermined hidden layer size is reached. At this stage OLS is halted and the updating of parameters phase is reached using EBP. In this version of CSF

training this next phase updates, first the output layer weights then secondly all hidden layer parameters (hidden weights, centres and angles) every epoch.

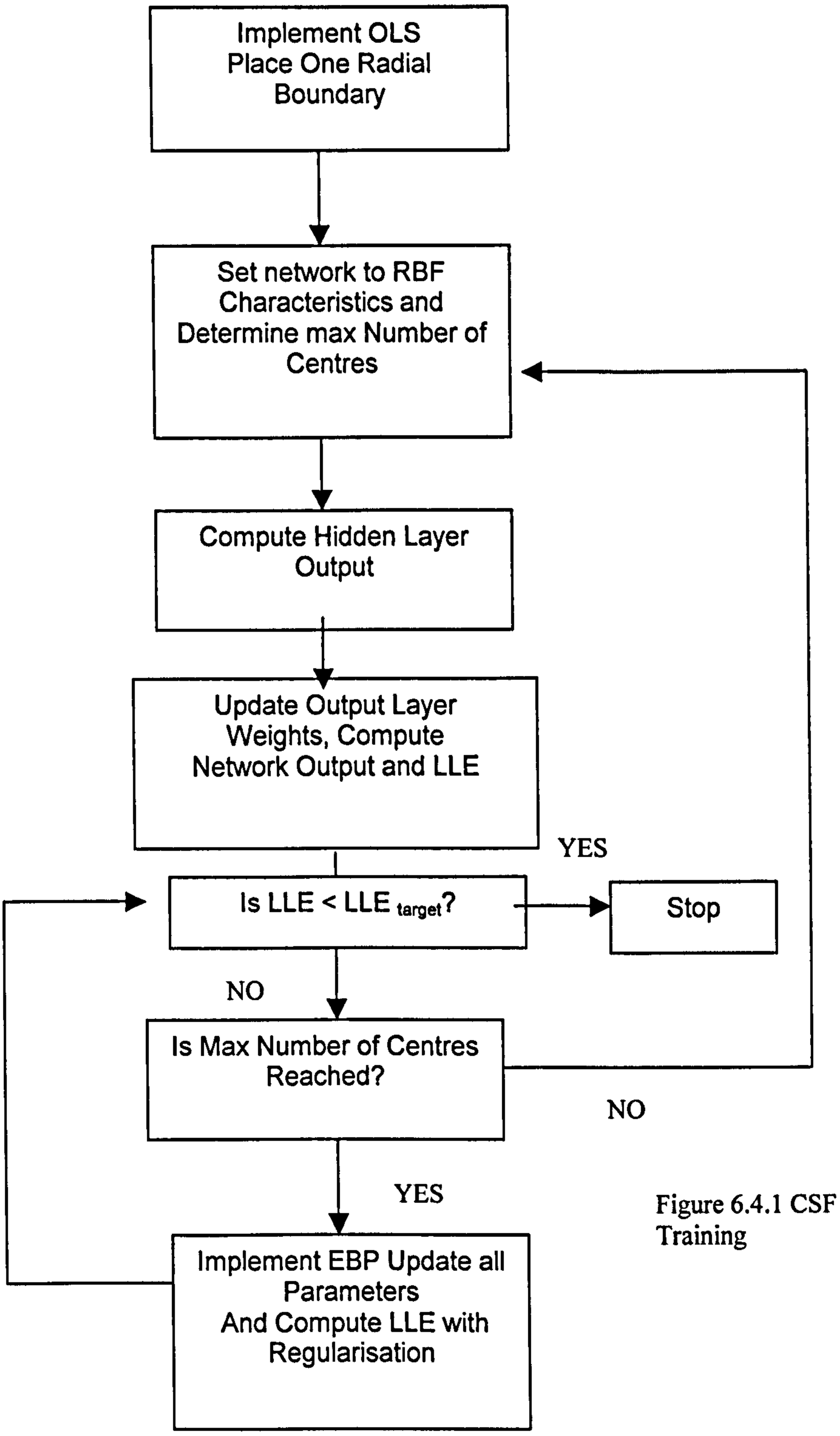


Figure 6.4.1 CSF Training

6.4.1 Twelve Fold Cross Validation

Validation Block
1:47

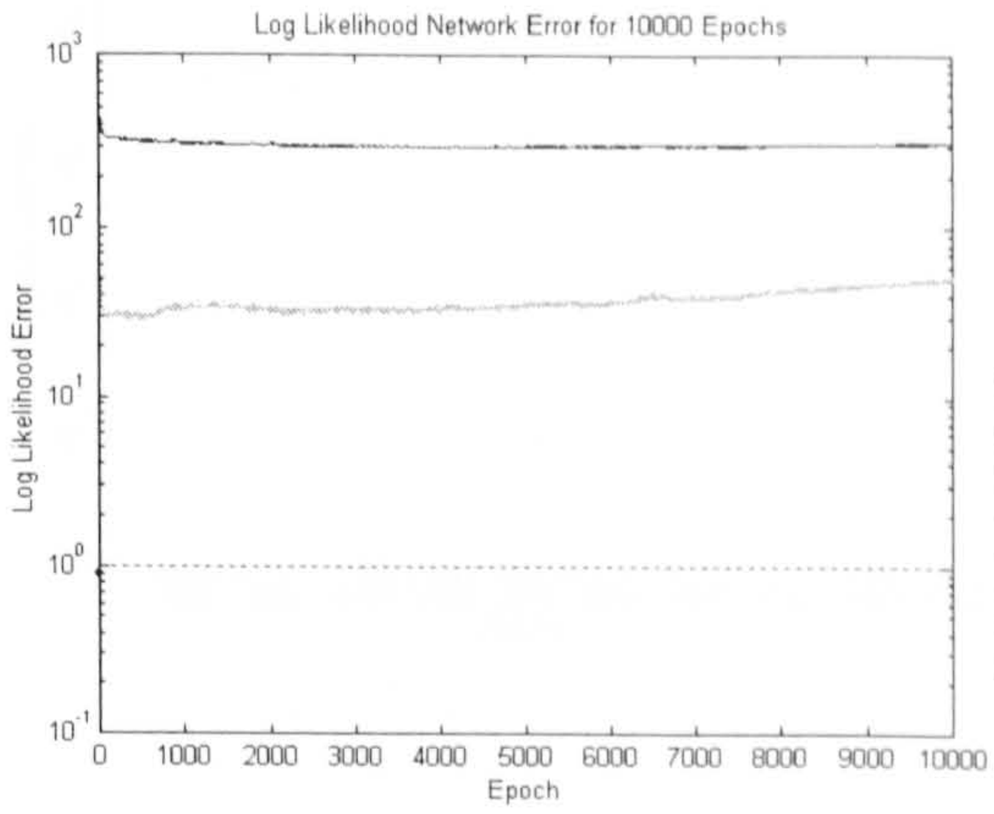


Fig 6.4.2

Validation Block, LLE	Training Block, LLE
50.54	311.224

Validation Block
48:94

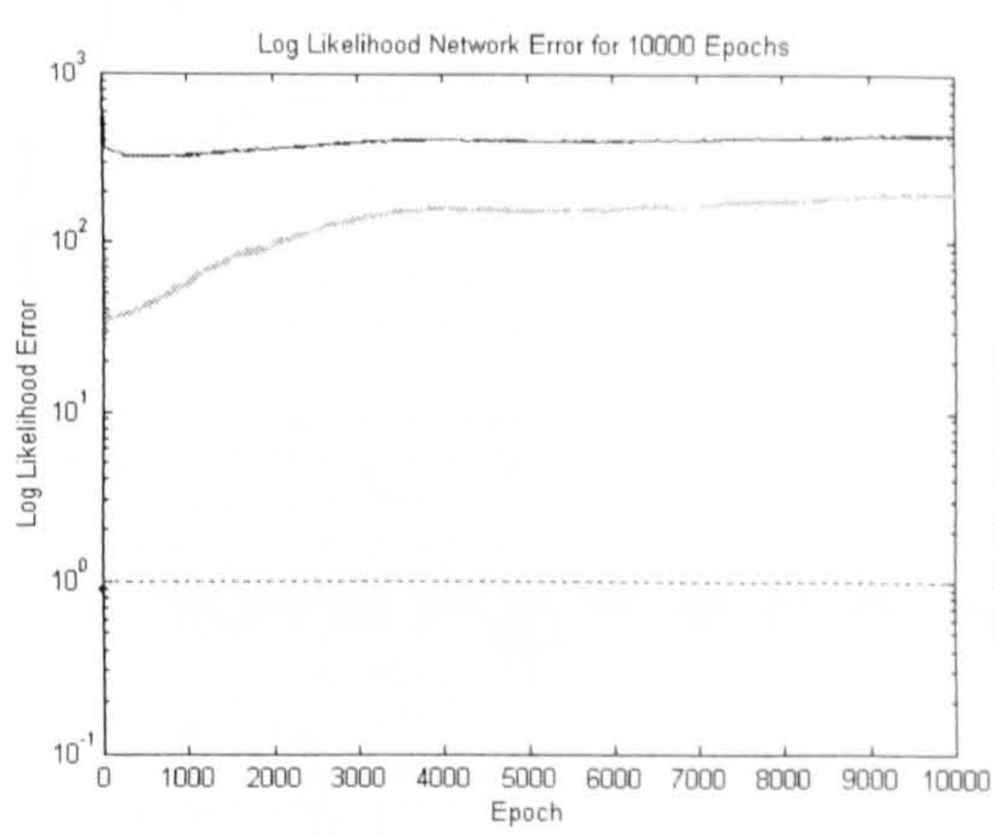


Fig 6.4.3

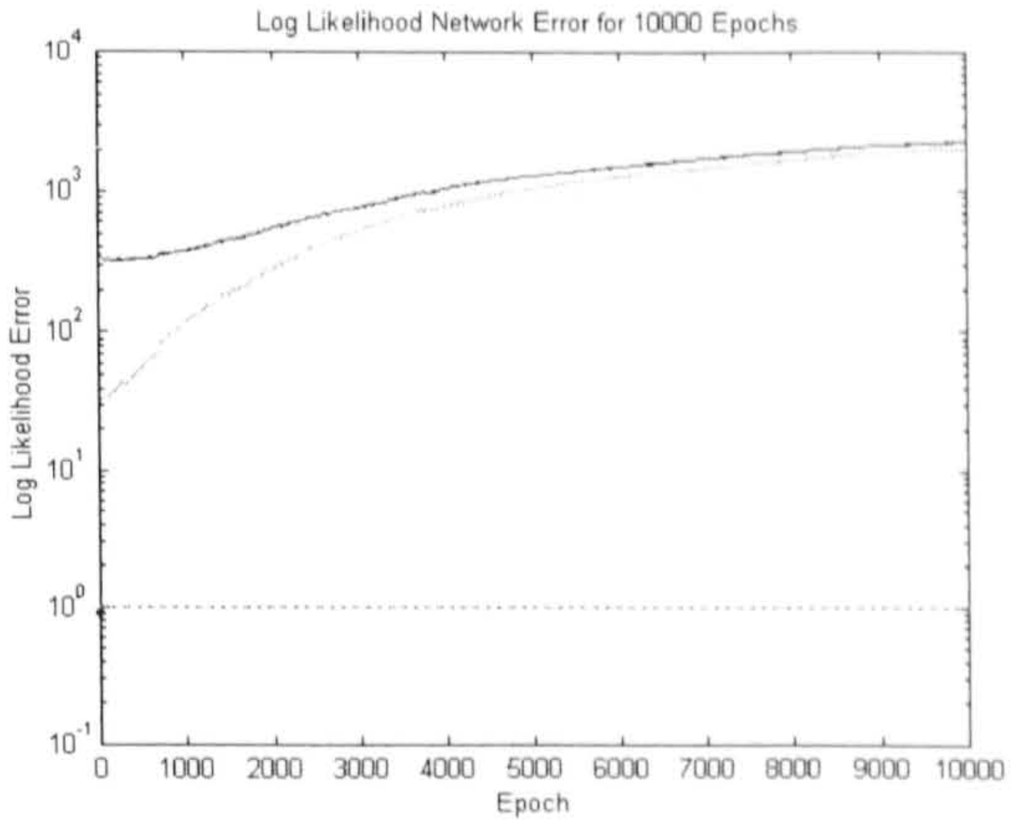
Validation Block, LLE	Training Block, LLE
196.02	442.066

Validation Block
95:141

Fig 6.4.4

Validation Block, LLE	Training Block, LLE
582.241	823.244

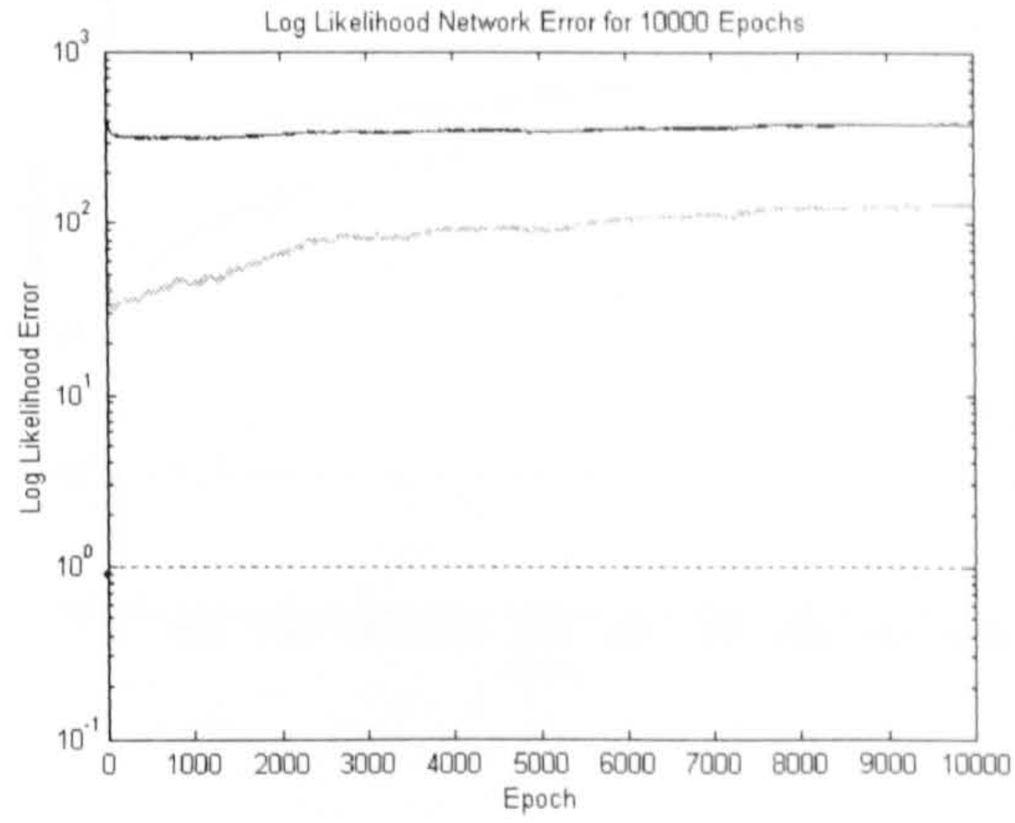
Validation Block
142:188



Validation Block, LLE	Training Block, LLE
2014.3	2251

Fig 6.4.5

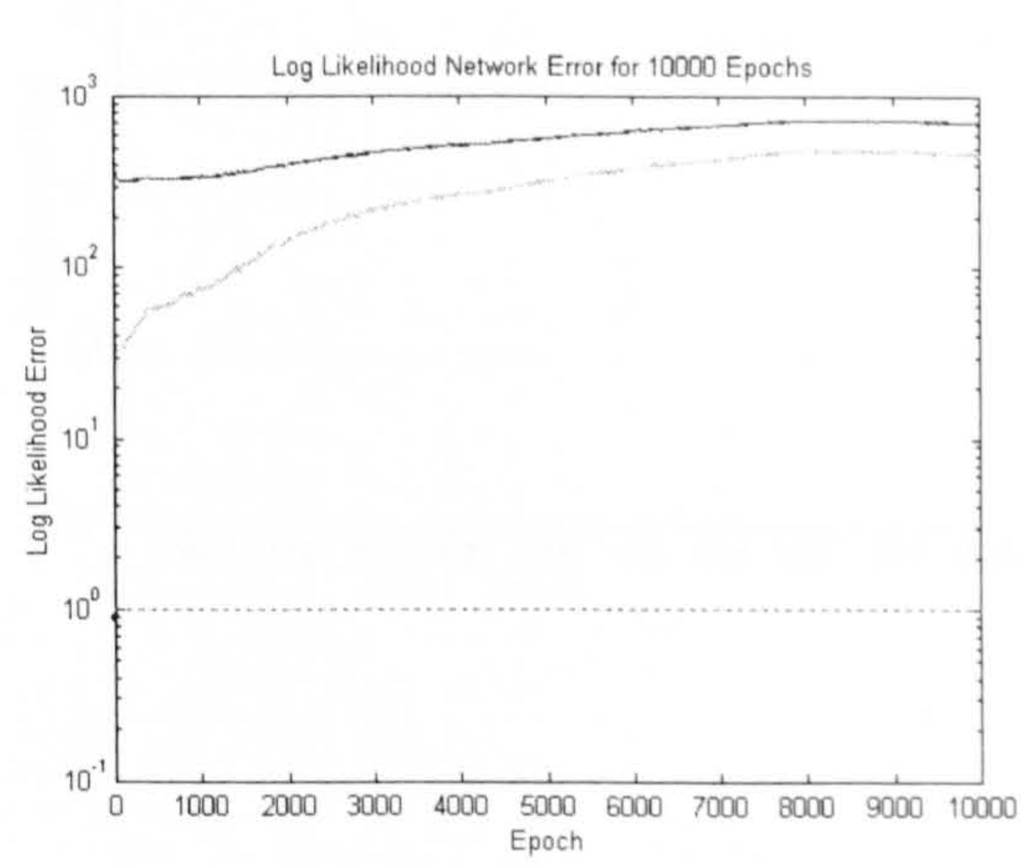
Validation Block
189:235



Validation Block, LLE	Training Block, LLE
131.4469	381.806

Fig 6.4.6

Validation Block
236:282



Validation Block, LLE	Training Block, LLE
456.06	700.89

Fig 6.4.7

Validation Block
283:329

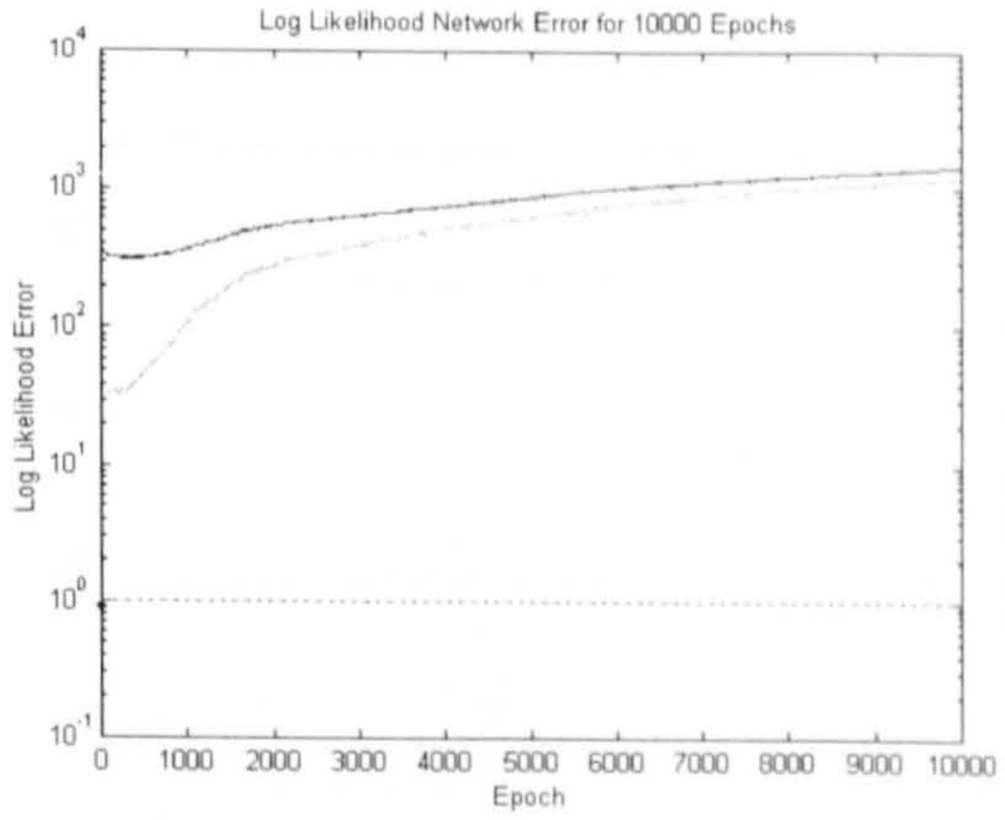


Fig 6.4.8

Validation Block, LLE	Training Block, LLE
1221.1	1457.88

Validation Block
330:376

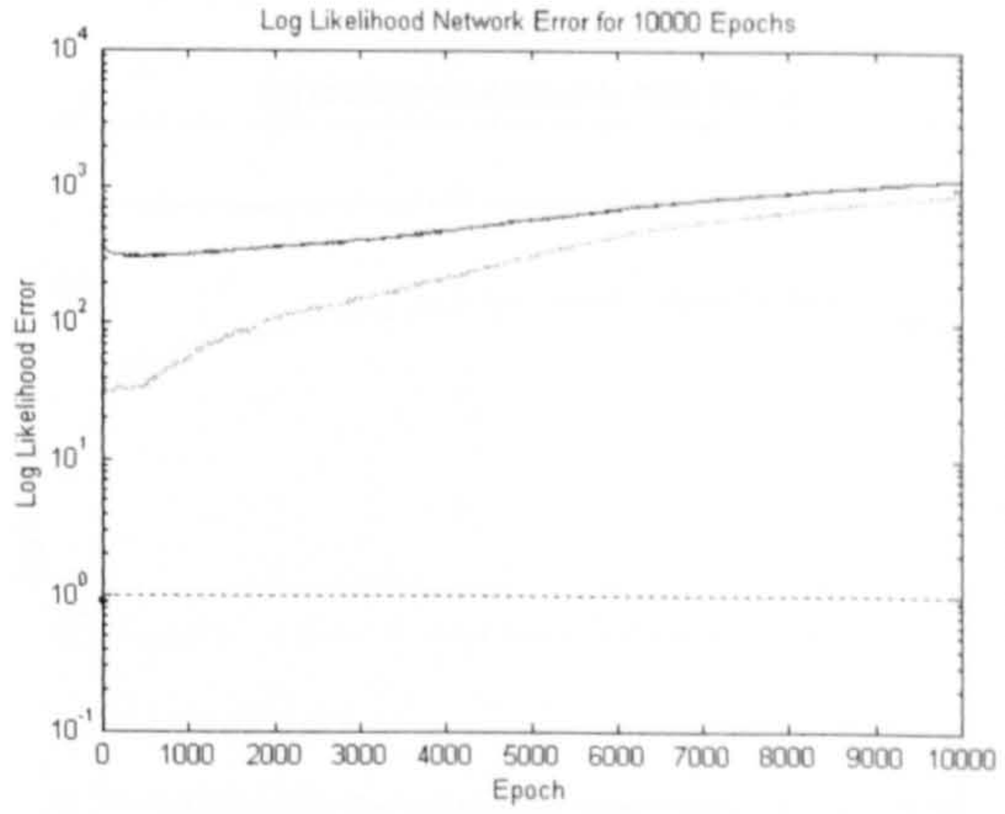


Fig 6.4.9

Validation Block, LLE	Training Block, LLE
907.02	1161.26

Validation Block
377:423

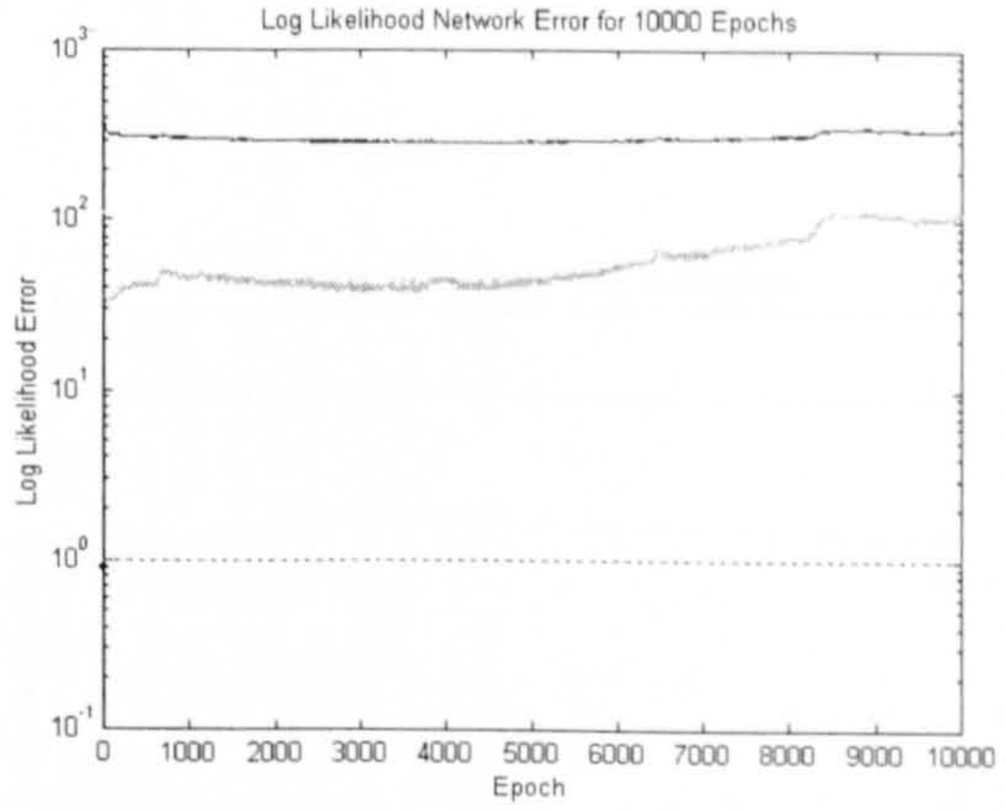
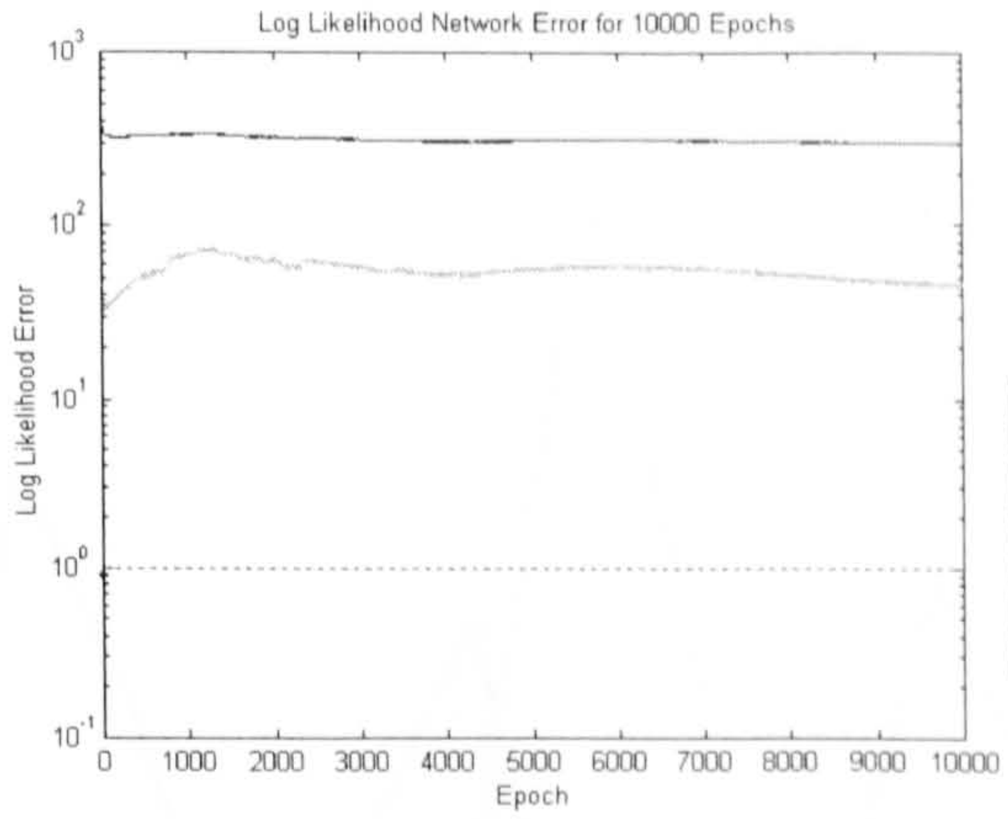


Fig 6.4.10

Validation Block, LLE	Training Block, LLE
110.299	347.171

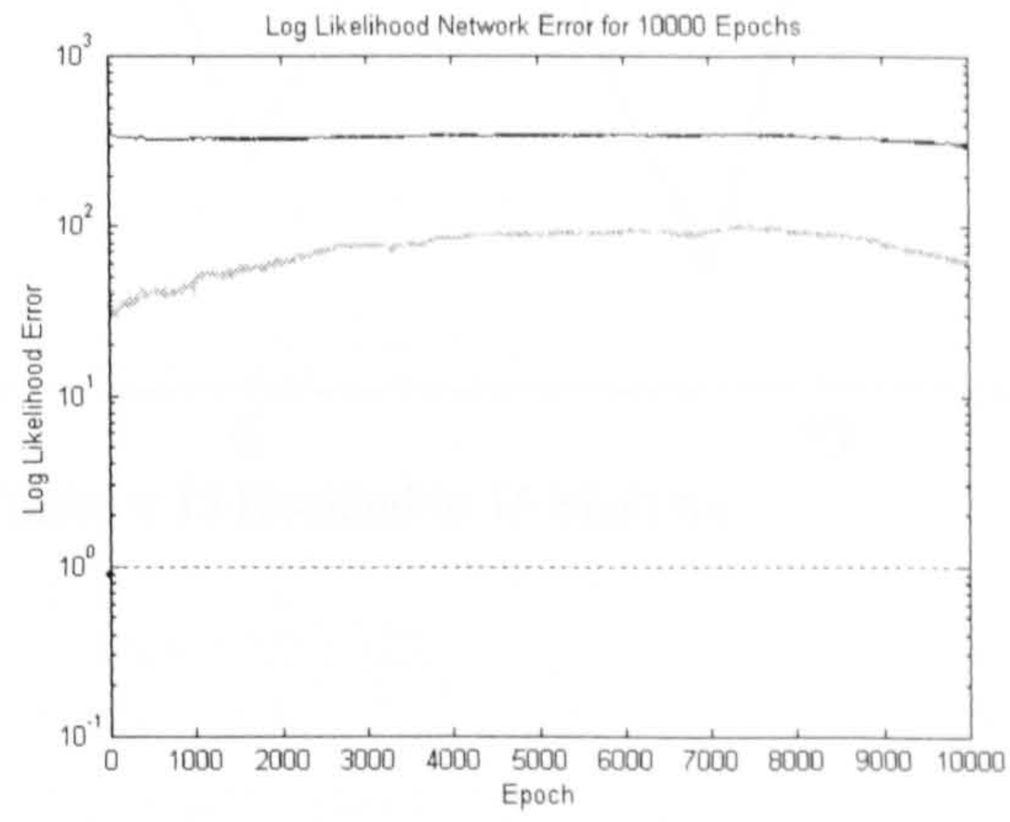
Validation Block
424:470



Validation Block, LLE	Training Block, LLE
45.37	292.8

Fig 6.4.11

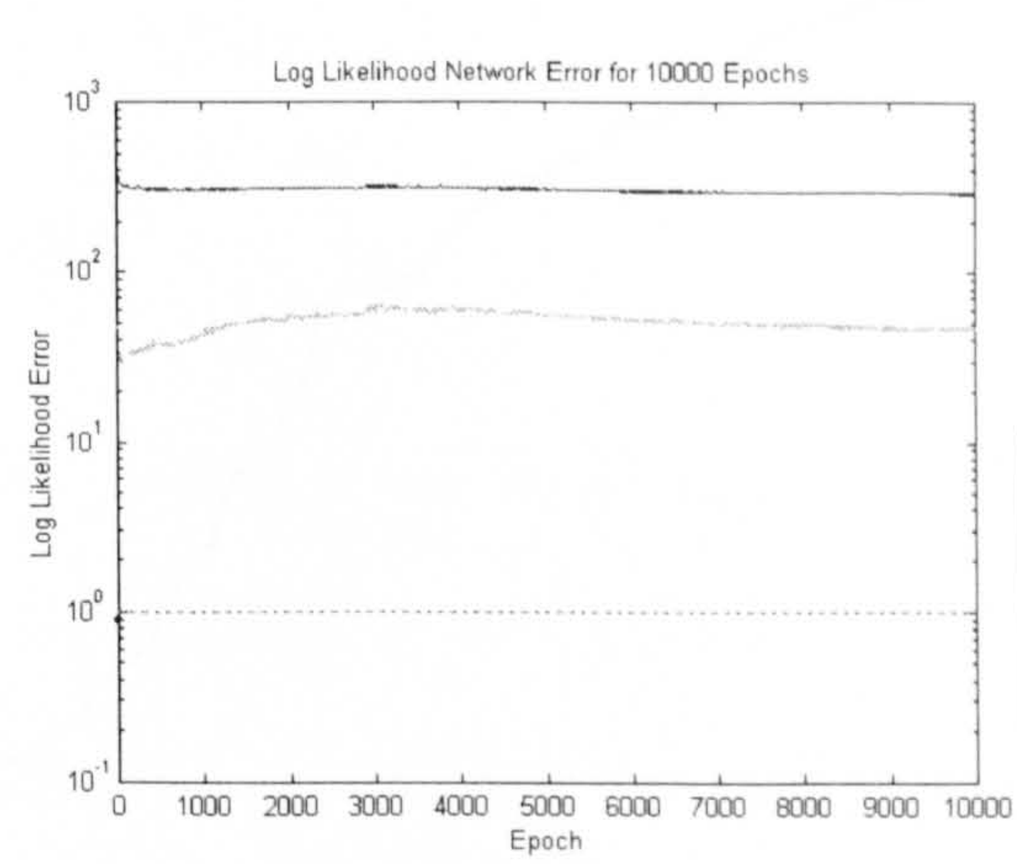
Validation Block
471:517



Validation Block, LLE	Training Block, LLE
65.616	313.348

Fig 6.4.12

Validation Block
518:564



Validation Block, LLE	Training Block, LLE
46.858	292.578

Fig 6.4.13

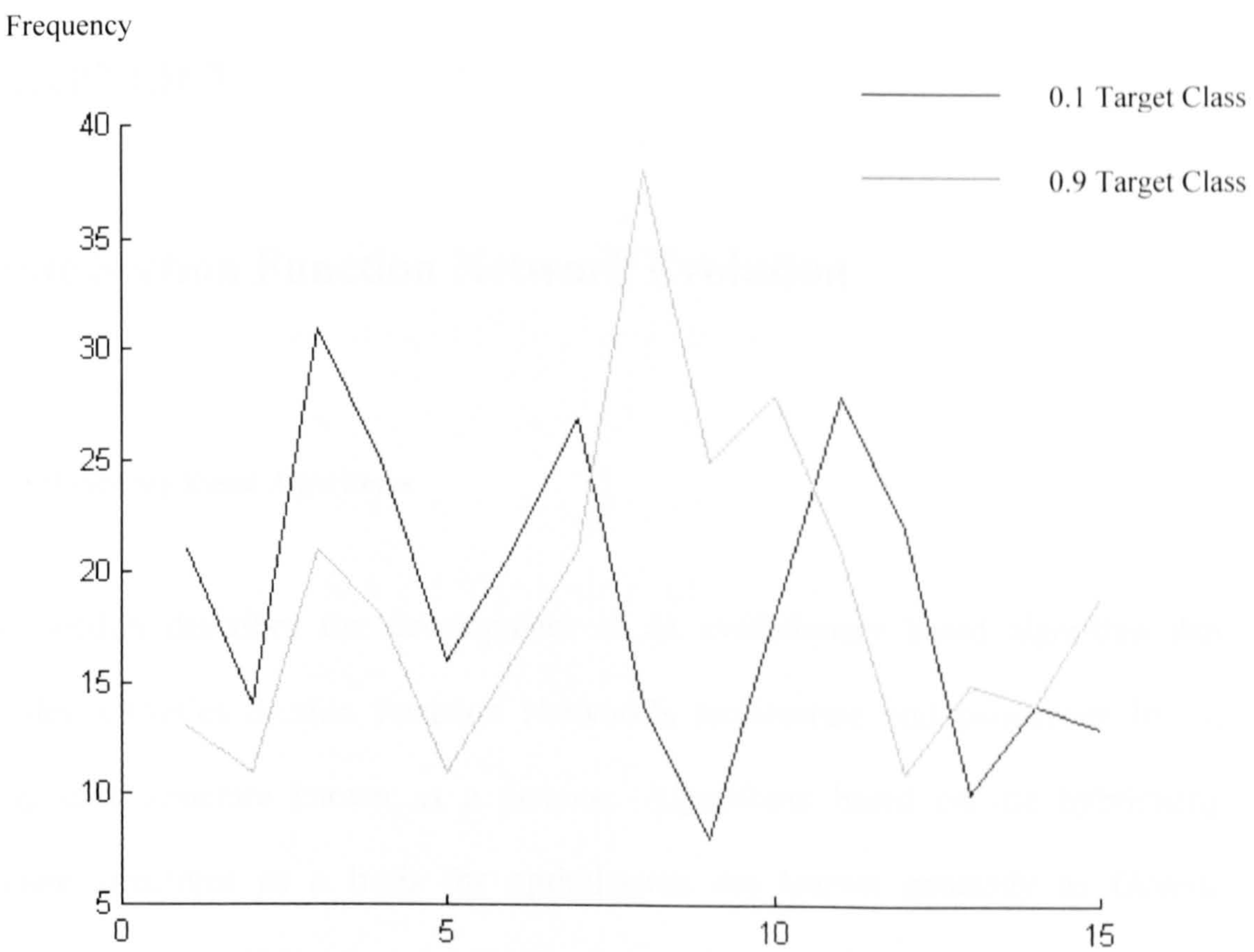


Fig:6.4.14 Histogram of output Values x 15 (totalled in 15 bins) for regular 12 fold cross validation

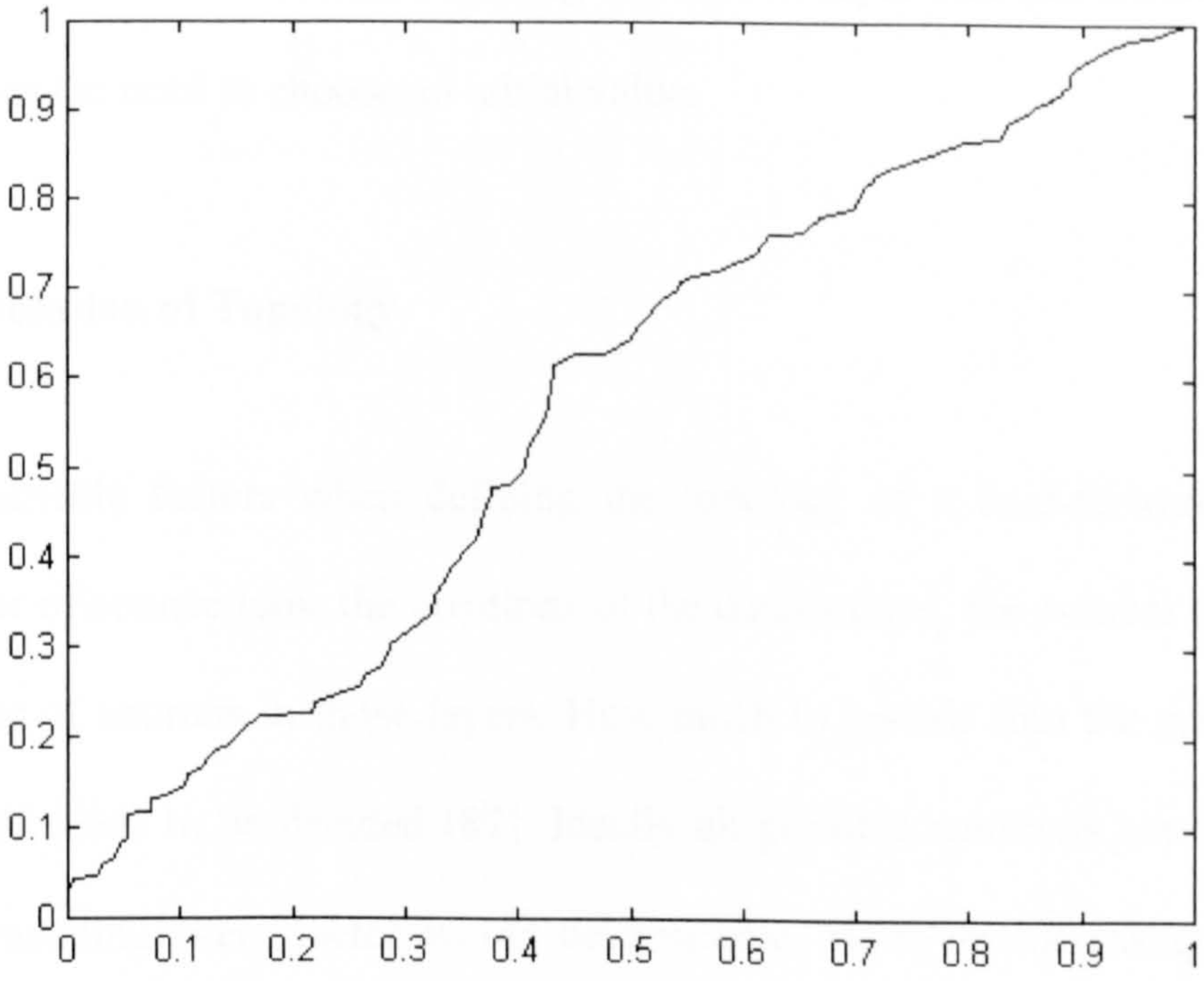


Fig 6.4.15: Receiver Operator Characteristic for regular 12 fold cross validation

CHAPTER 7

Conic Section Function Network Evolution

7.1 Evolutionary Based Algorithms

This section describes the development of an evolutionary based algorithm that encodes a Conics Section Function Network's architecture and parameters into a string data structure known as a *genome.* Algorithms based on the hybridising genome structures as a basis for optimisation are known generally as *Genetic Algorithms* (GA) [35]. Such methods though known to be slow can increase generality due to the possible removal of predetermining parameters that inherently constrains the search space, for example including choosing hidden layer size before the commencement of EBP. Placing the hidden layer size parameter in the *genome* removes the need to choose an initial value.

7.2 Evolution of Topology

The variable factors when defining the topology of a feed-forward ANN are the number of connections, the positions of the connections, the number of layers and the number of neurons in those layers. How much to encode into the genome for use in the GA needs to be decided [87]. Ideally all possible solutions are sought after and thus encoding every factor would be desirable. However increasing the number of factors that are simultaneously being crossed over and mutated in the GA makes the possibility of successful evolution lower. For this GA the encoded factor is the

number of hidden neurons. This leads to all generated networks having one hidden layer and being fully connected. Hence the outcome of the topology part of the evolution is simply a positive integer. The complexity of CSF networks lies not in their topologies but in their internal parameters however the hidden layer size is an important topological factor. The method for encoding the hidden layer size can be done by reserving a section of the genome for the evolution of this integer. Ideally this number should be unlimited, that would however cause decoding problems since the GA needs to recognise where the topology genes end and the parameter genes start. If say the maximum allowable number of nodes is 32 then the first five genes can be reserved as binary digits.

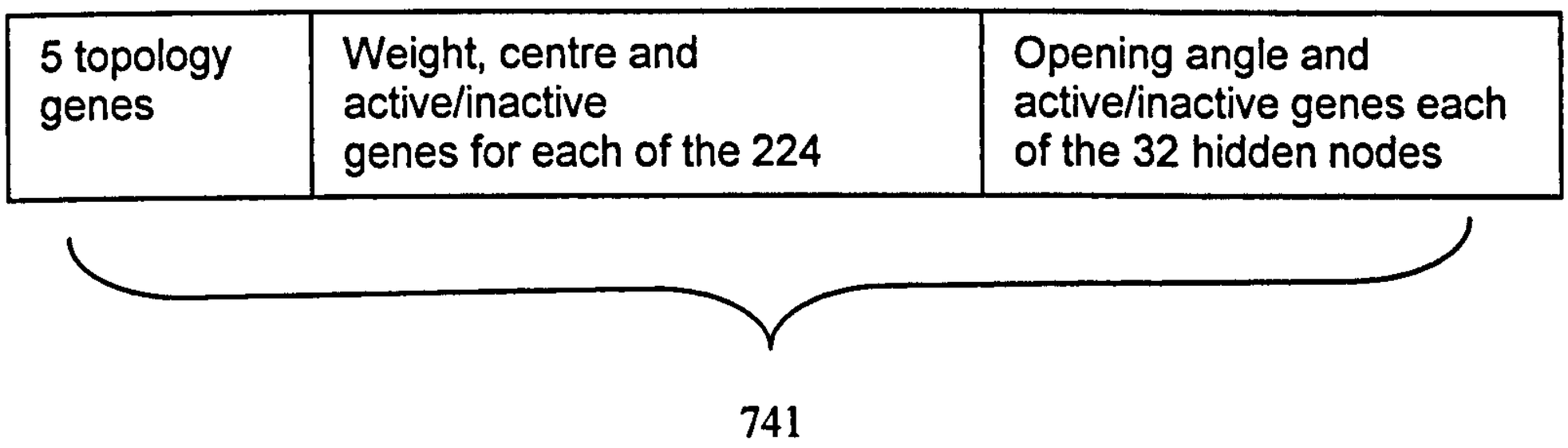
7.3 Evolution of Parameters

One of the main differences between CSF and other feed forward networks is that the CSF assigns two parameters to each hidden connection (a weight and a centre) and not one [25]. This doubled number of parameters gives rise to the consideration of real number encoding of the genome (each gene is a real number). Doing this serves two purposes; first it reduces the length of the genome and secondly it helps to keep the real value of the parameter and discreteization from binary gene encoding. From the topology genes the number of parameters that has to be included is defined. The GA can recognise which parameters to include by the use of an active/inactive bit prior to each set of parameters within the chromosome. This bit appears before each weight and centre value, for every connection, and before every opening angle value for every hidden node. This is done to include all the parameters for future generations and to maintain equal overall genome length.

7.4 The Genome Structure

The overall length of the genome depends on the maximum hidden layer size, the input dimensions and the output dimensions. For example if the maximum hidden layer size is 32, the input dimensions 4 and the output dimensions 3 then there must be 224 connections accounted for. Therefore the length in this case is 224x3 + 32x2 + 5 making a genome with 741 genes.

Figure. 7.4.1 Genome Make Up



7.5 The Genetic Operators

The Genetic Algorithms uses three main operators reproduction, crossover and mutation. The reproduction operator is a selection procedure based on the fitness measure of each of the individual strings. The probability that a string is selected, for the next stage in the GA, is proportional to its fitness level. This is a common selection procedure also known as *roulette wheel* selection [35]. Once two 'parent' stings are selected the crossover operator cuts both strings at the same point and swaps the two sections of the strings that follows the cut point.

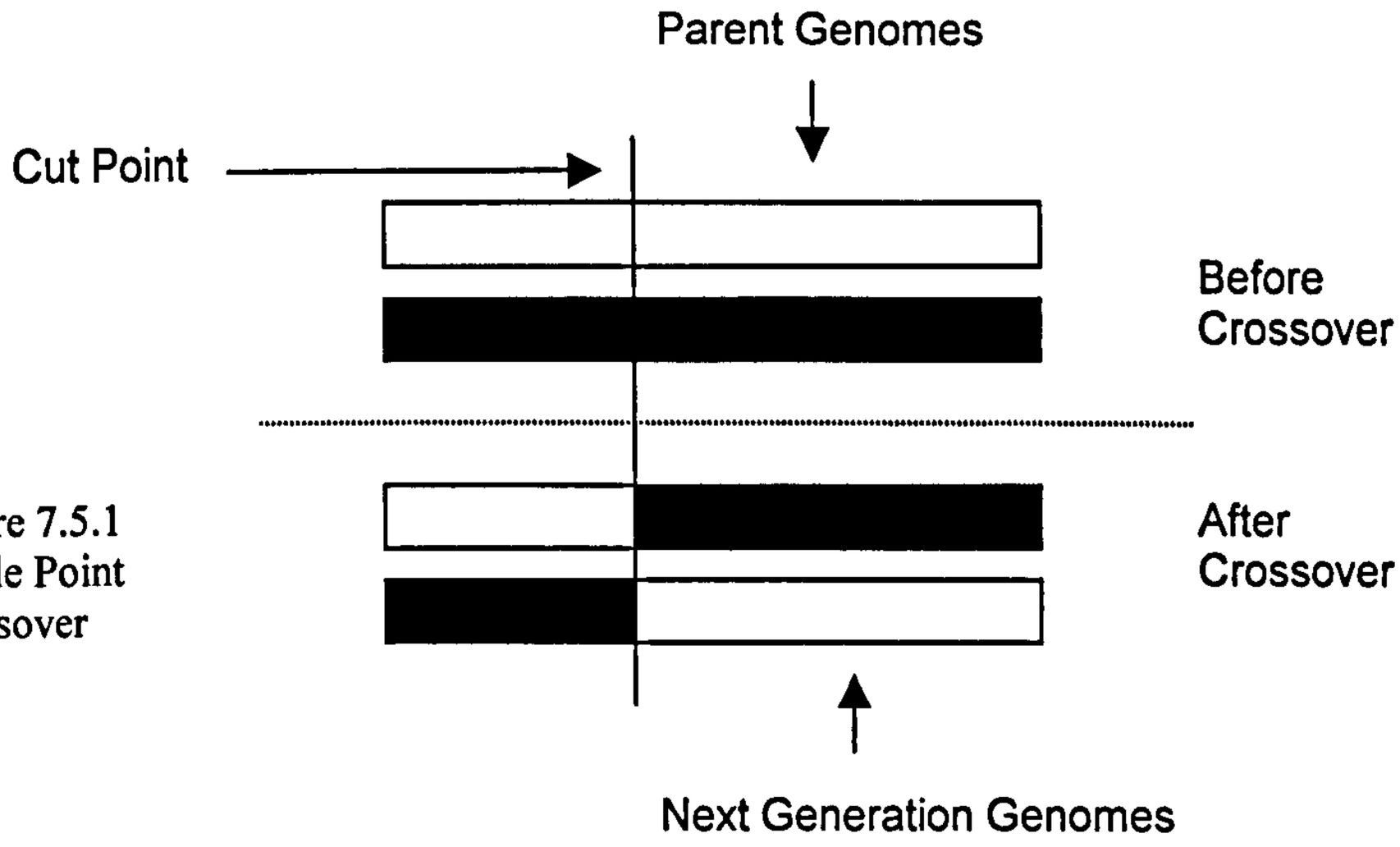


Figure 7.5.1
Single Point
Crossover

By translating the string into a CSF network and applying input patterns the fitness of the chromosome can be evaluated. The difference between the acquired outputs from the CSF network and the desired outputs yields the error margin the inverse of which can be used as a fitness measure. The higher the fitness of the individual the more likely it is to be selected in the 'roulette wheel' selection scheme. This scheme generates a random positive integer from a set. This number selects an individual, however the fitter individuals have more integers from the set assigned to it, hence increasing their chances of selection. Alternative to the roulette wheel selection is elitist selection where the fittest cases are automatically chosen for crossover. The Mutation operator alters the gene by a random change Δx if x is the value if the gene. The likelihood of this change being implemented is set to a uniform percentage, usually a small probability (1%).

7.6 Implementation and Testing

The development for the algorithm aims to keep the GA as simple as possible, this is because in general GA's are notoriously slow as compared to alternative optimisers.

A simple test function is implemented. This function simply sums all the variables in the genome and deems the largest as the best.

This first GA implemented an elitist selection procedure, single point crossover and random uniform mutation. Ten stings containing 700 real numbers ranging from -5 to5 (this is a typical ANN weight range) were randomly generated to initialise a population. The two fittest were chosen for crossover (elitist selection) and the remaining eight in the population were regenerated for the next generation. One crossover point was randomly selected and the fitness of the two offspring are evaluated. The following diagram describes this particular test algorithm in more detail.

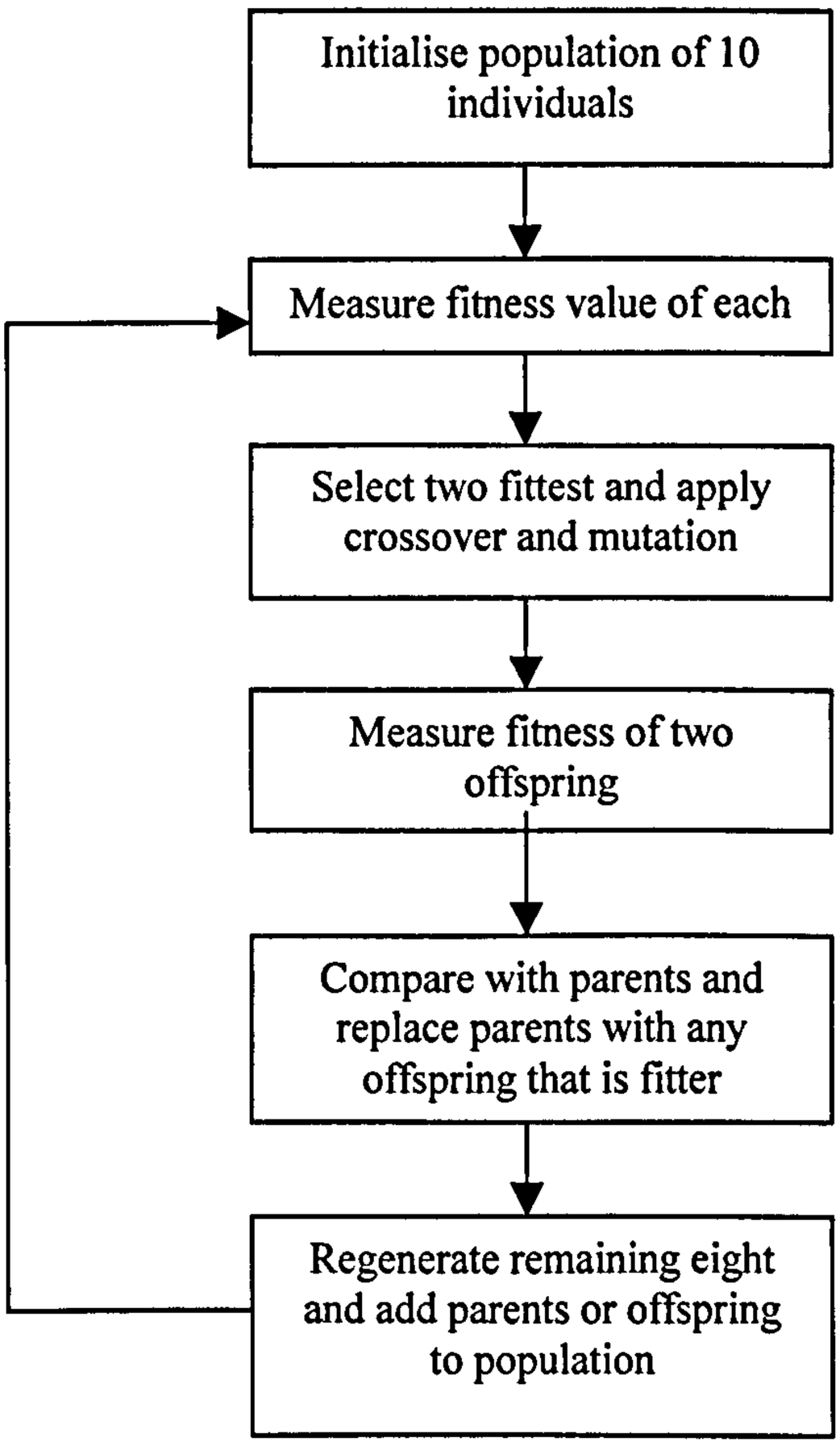


Figure 7.6.1 Test 1 GA

The following diagram demonstrates typical performance of this algorithm.

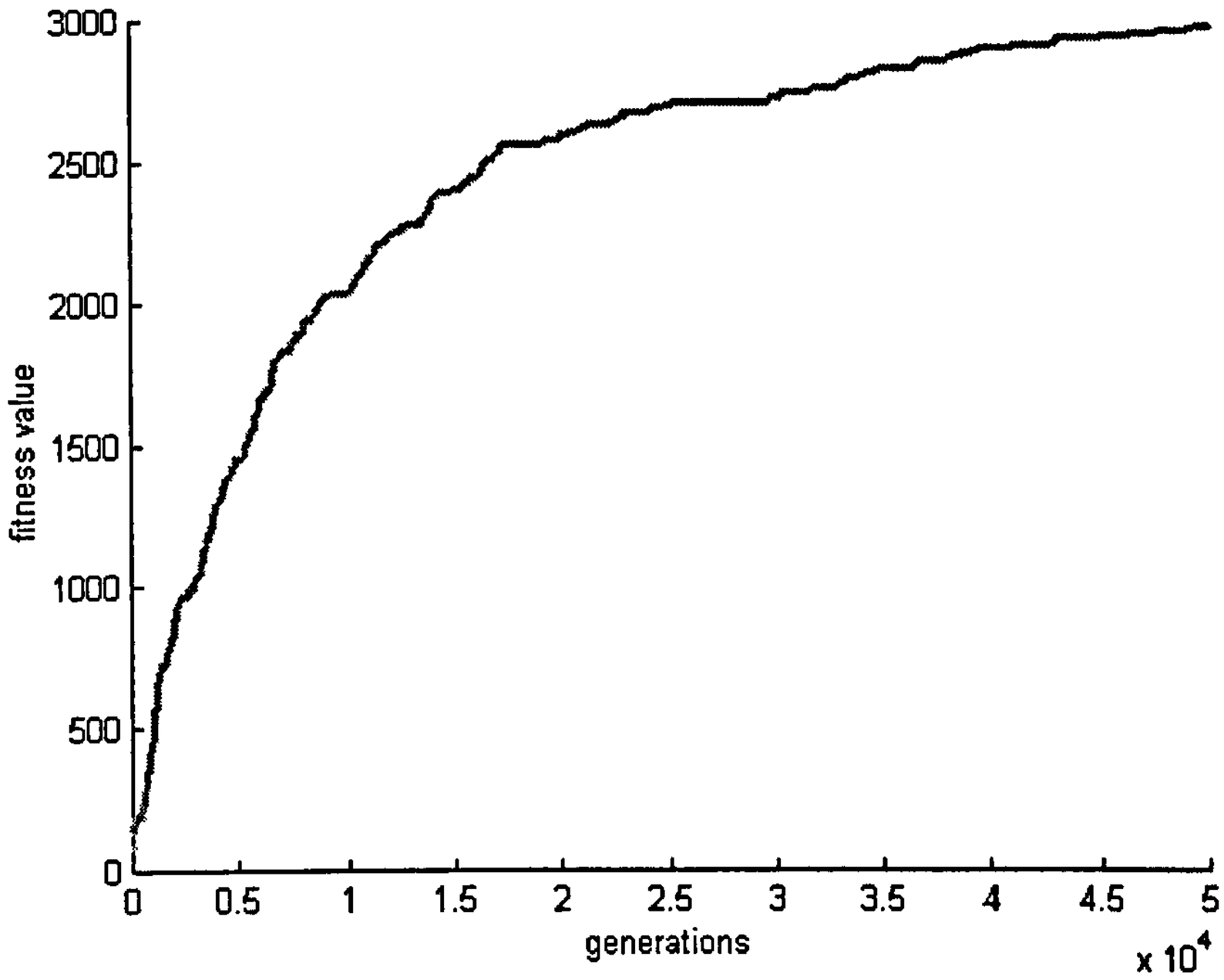


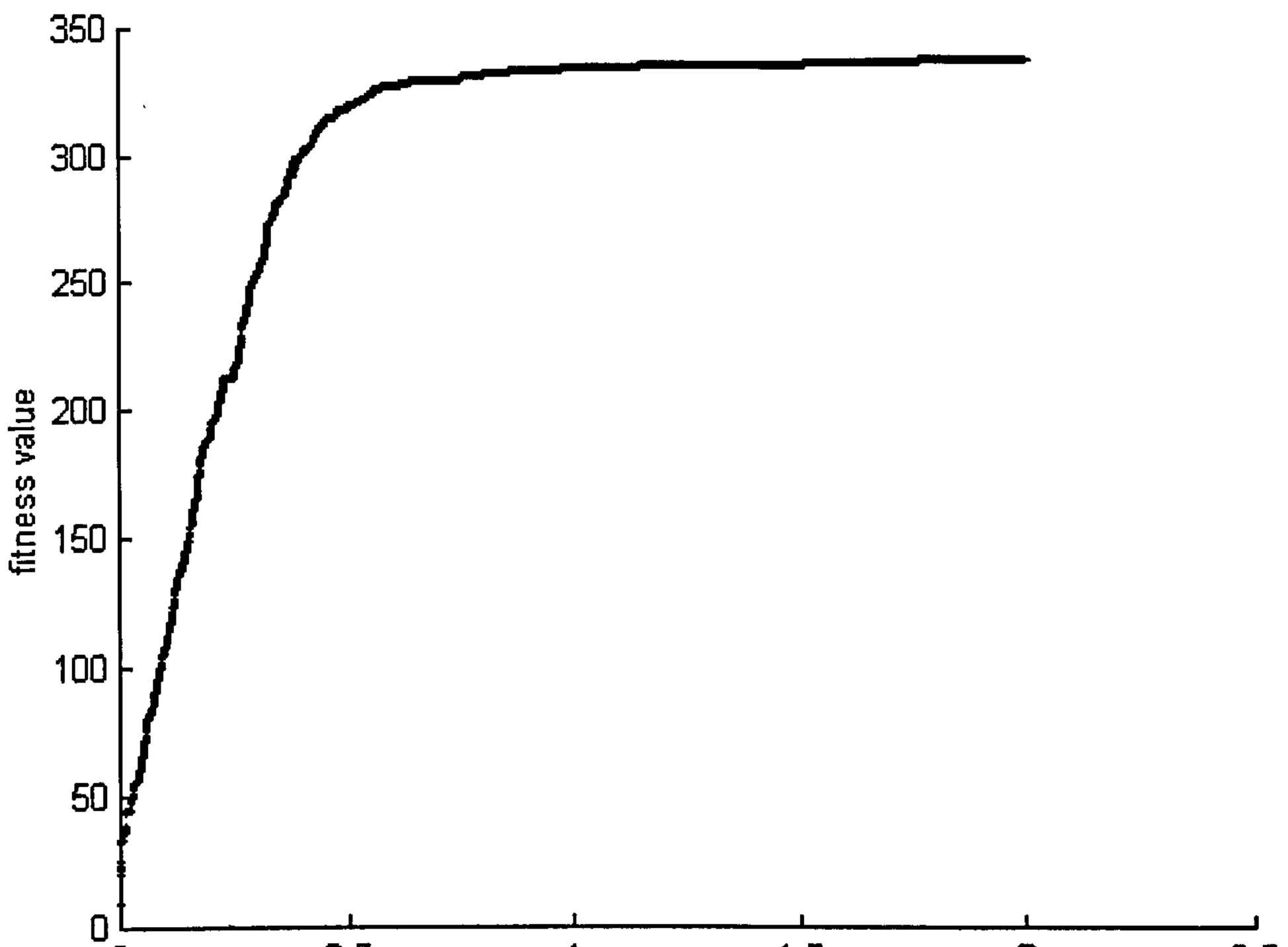
Figure 7.6.2 Test 1 GA Result

The maximum fitness value obtainable in this test function is 3500 (all 700 elements in the genome reaching 5), this algorithm typically stops converging at the 3000 level. The probability of each element mutating was set at 1%. If the mutated element exceeded 5 or became less than -5 then the effects of the mutation was cancelled instead of setting the result at the limits. The reason for this is so that the algorithm is not biased in maximizing numbers, since that is the test scenario.

Roulette selection is also tested instead of elitist selection, however is found not to converge. This may be due to the large size of the genome and consequently the small difference between the most fittest and the least fittest. Even with the fitness

values squared and cubed the algorithm did not successfully select the better individuals for the next generation and hence did not converge toward the 3500 mark. A second test algorithm that evolved genomes of length 68 instead of 700 is also considered. This is done to test the performance on smaller search spaces. Previous experiments [58] have shown that CSF networks can train, using conventional algorithms, with 5 hidden nodes for the classic Iris Plant data set [32], CSF is found to successfully train [57][58]. For this GA to encode a 5 node hidden layer CSF network it requires 68 elements (20 input weights, 20 centres, 5 hidden biases, 5 opening angles, 15 output weights and 3 output biases). The maximum fitness in this case is 340. The performance of the algorithm in this version is shown in the following figure.

Figure 7.6.3 Test 2 GA Result



This converges directly to the desired limit. With this result the next step was to replace the dummy fitness function with a CSF network. The fitness function for this case is:

$$f = \frac{1}{\sum_{n=1}^{75}\sum_{m=1}^{3}(t_{n,m}-o_{n.m})^2} \tag{7.1}$$

Where f is the fitness function, t is the desired target from output node m, o is the actual output and n is the number of input patterns given.

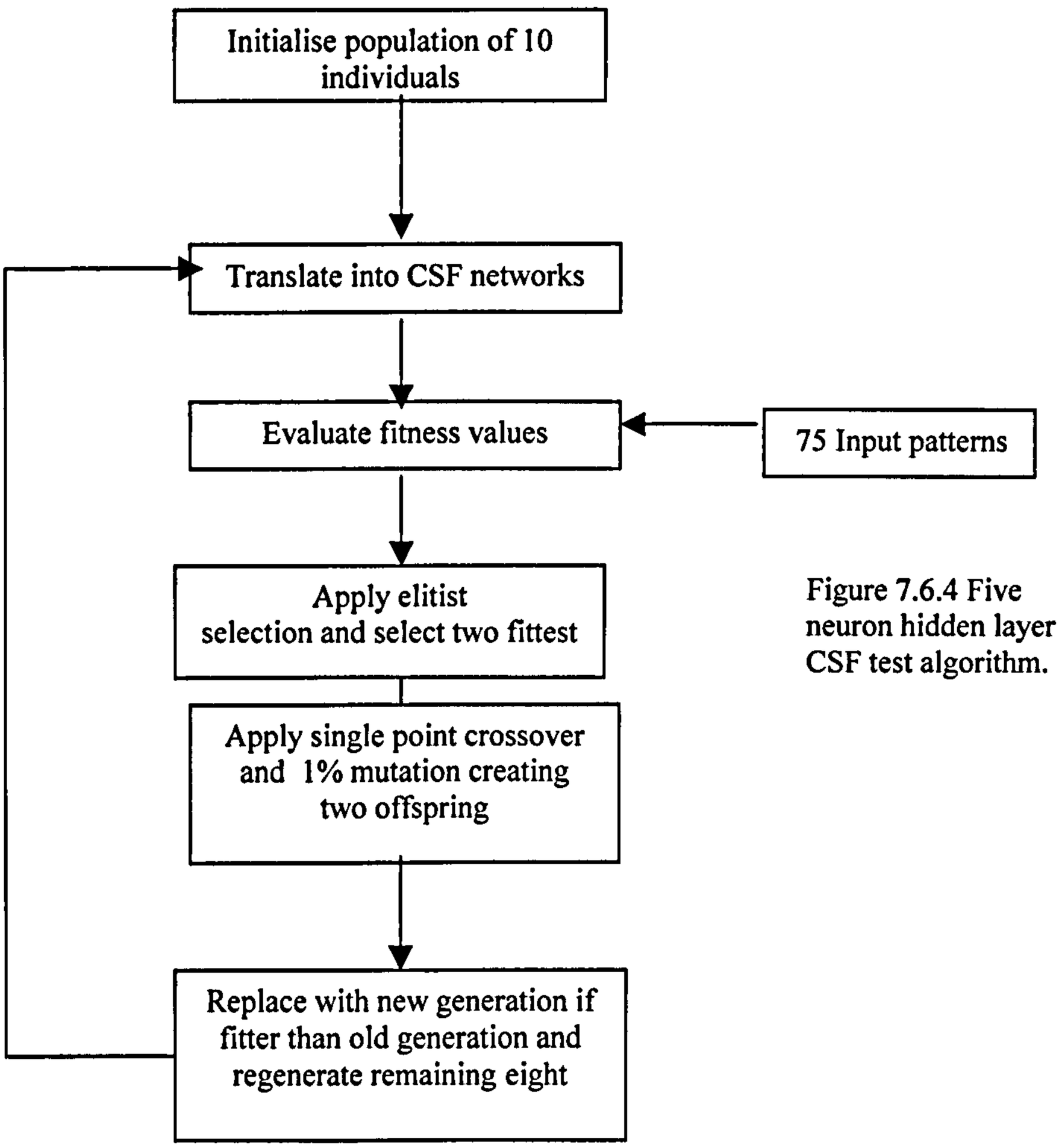


Figure 7.6.4 Five neuron hidden layer CSF test algorithm.

The following figure demonstrates the performance of this algorithm on the CSF test function. In this case the fitness has no limit, the higher the value of f the less error there is between the desired and actual outputs of the CSF network:

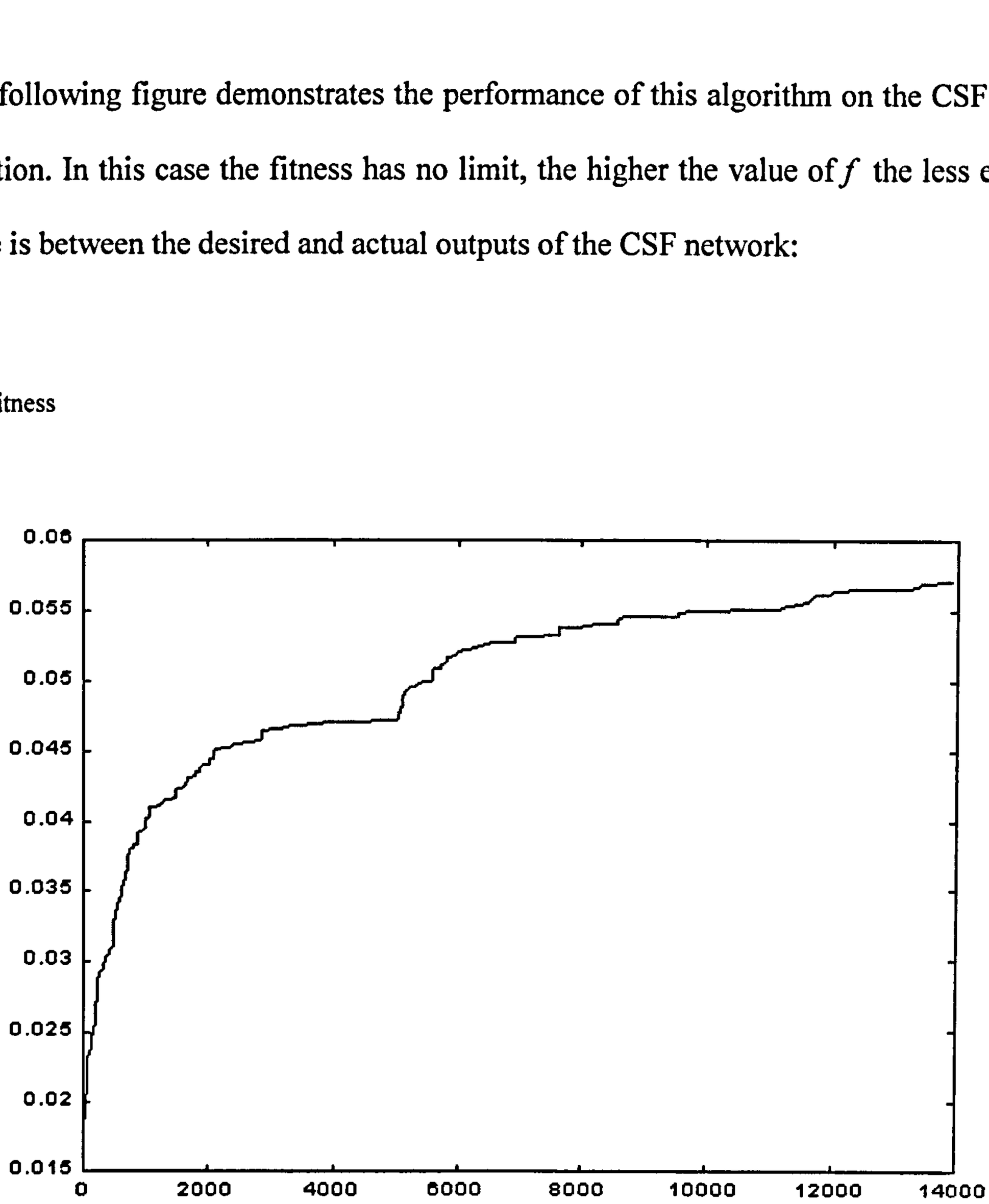


Figure 7.6.5 GA result for CSF Simulation on Iris Data

The initial fitness in this case is 0.015 translating to a margin of error, for 75 samples from the Iris Pant data set, to be 67. The highest fitness value obtained in this test is 0.0557 translating to an error margin of 18.

7.6.1 Multi Point Crossover

Genetic Algorithms can also be implemented using more than one cut point in the crossover operation. This is called *multi point crossover*, figure 7.5.1 demonstrates single point crossover where the next generation comprises of one part of each parent. If say the number of cut points is three then three positions are randomly selected on the chromosome length. Adjacent sections of the genome are then swapped.

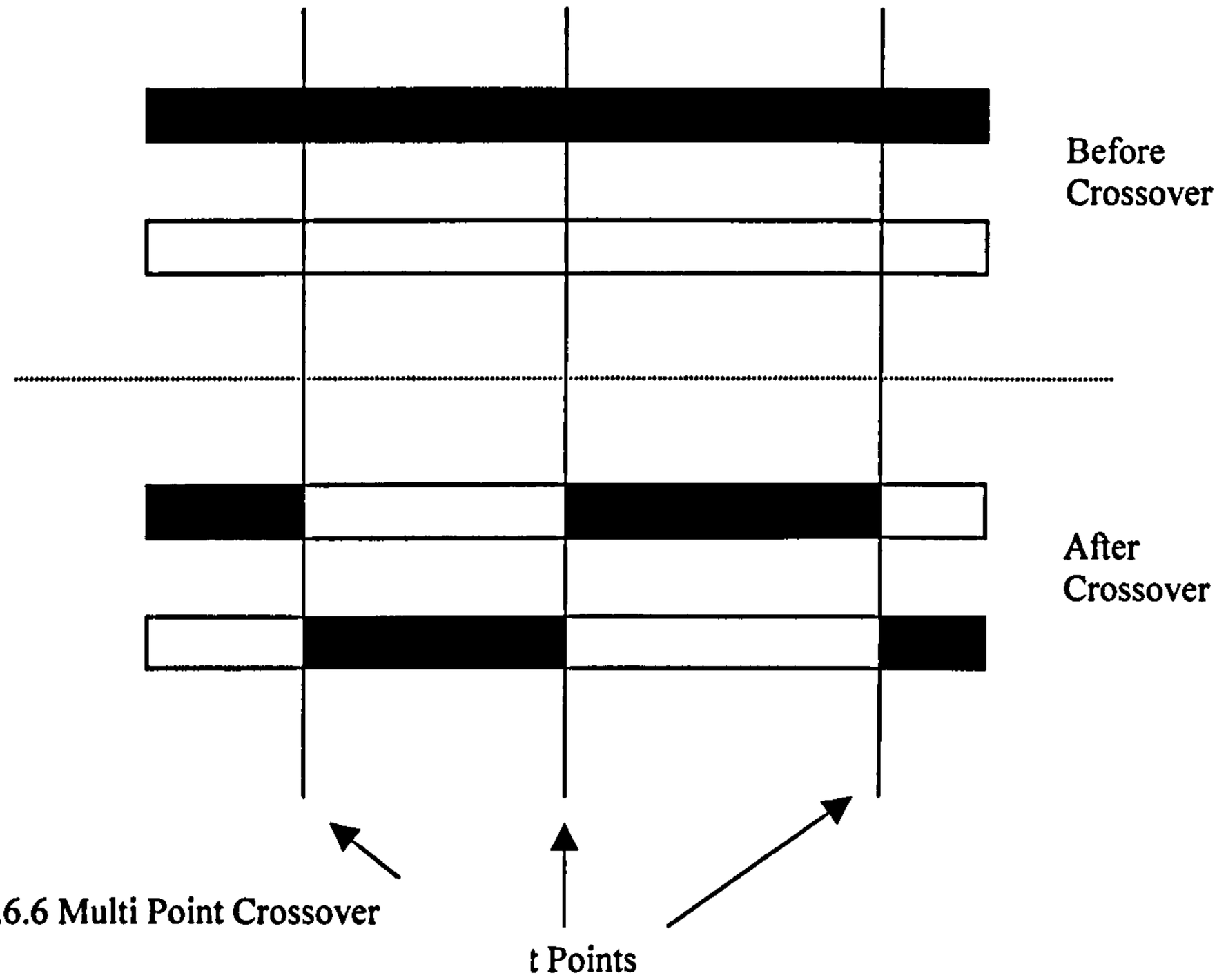


Figure 7.6.6 Multi Point Crossover

The number of cut points can either be set to an optimal constant or it can be dynamic.

Since the initial population all have similar fitness values the number of cut points can be set to a high value at the start of the GA. This will mix the genomes more than lower numbers of cut points. As the GA progresses the number of cut points can be lowered as not to lose useful genes, especially when used in ANN's where good weightings are only good as a set of weights and not as single weight values. The CSF

network is prone to this problem more so than other ANN's since CSF complexity lies in it's parameters and not the number of nodes.

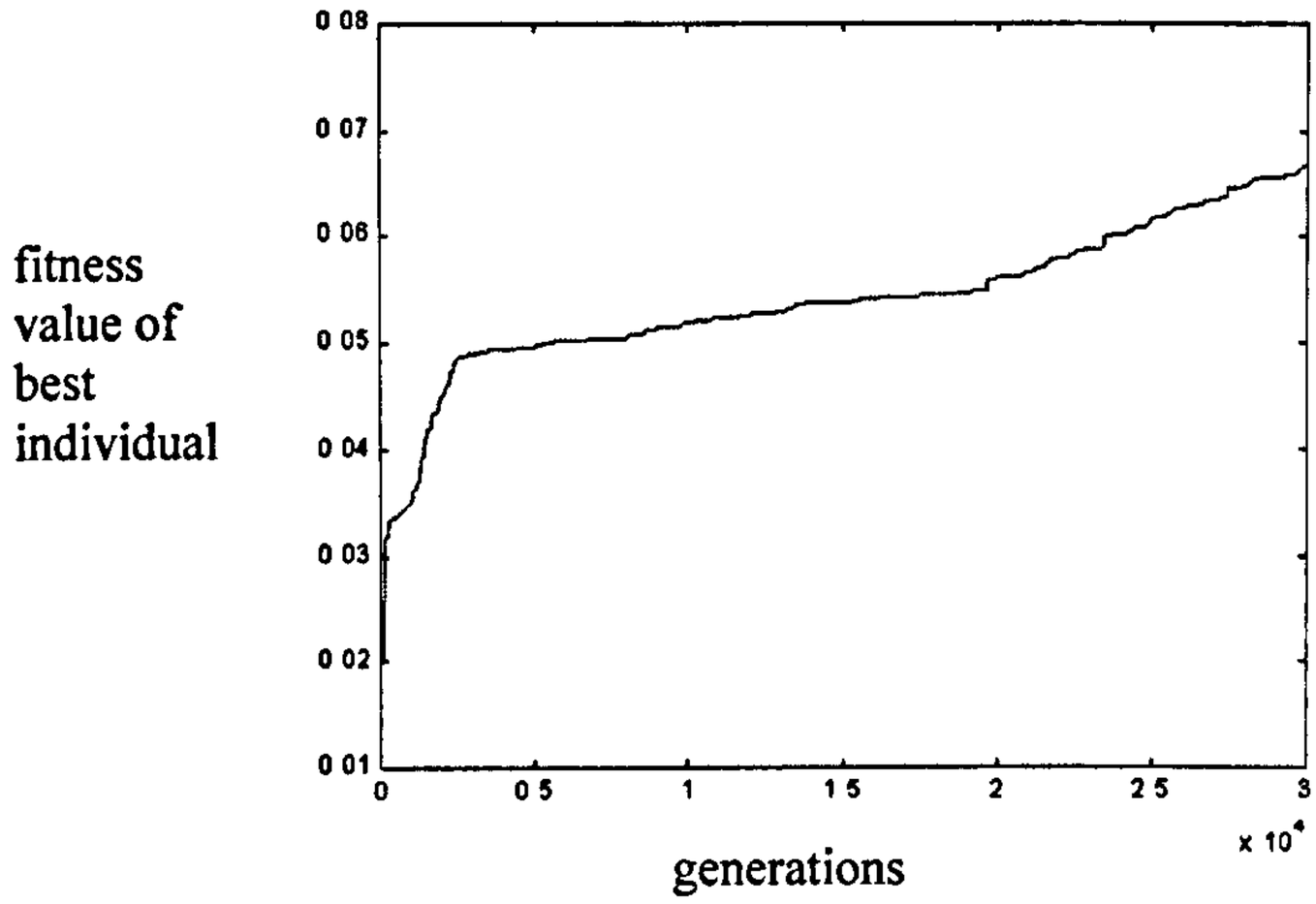


Figure 7.6.7 One Point Crossover

2 point crossover

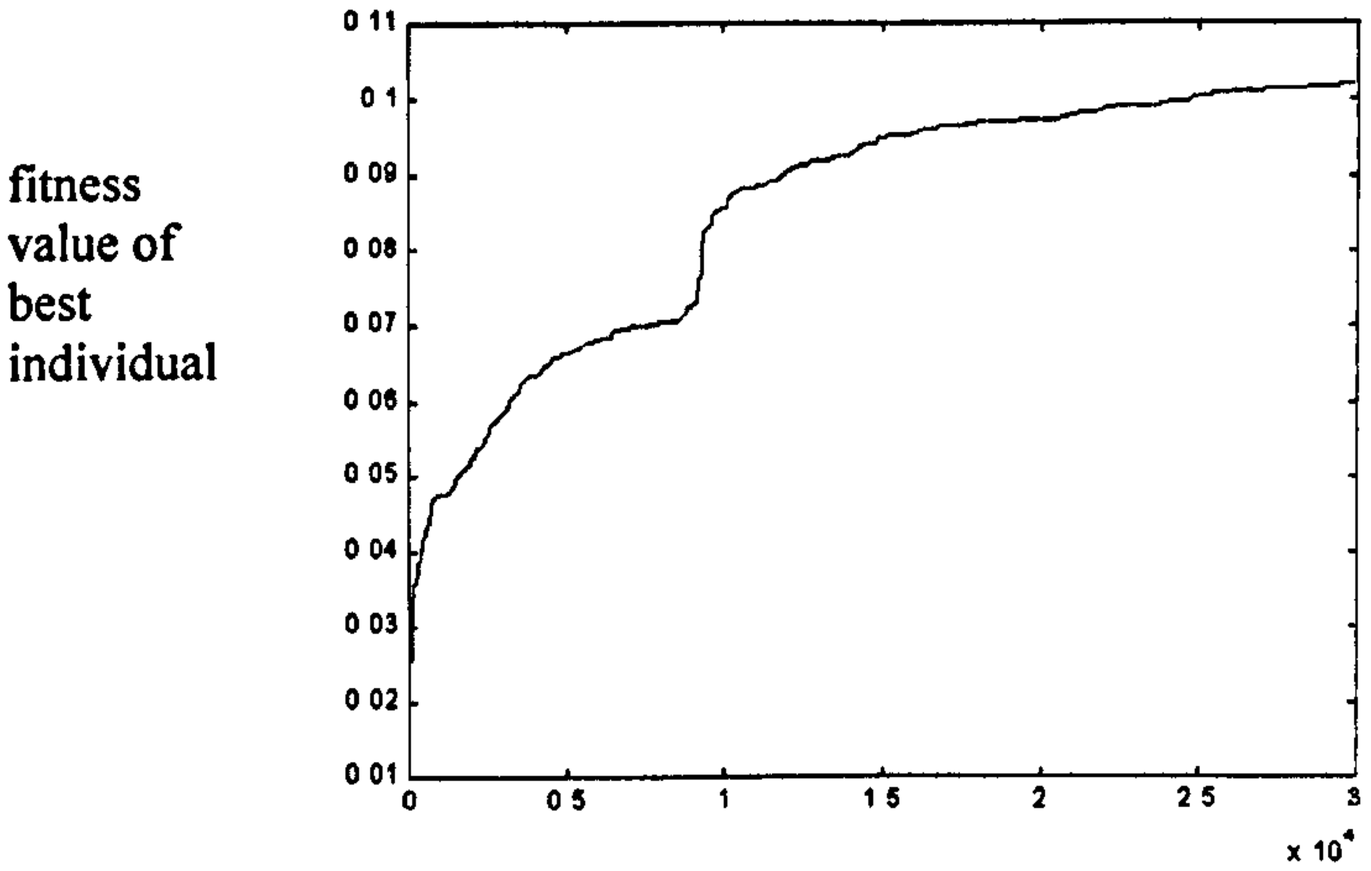


Figure 7.6.8 Two Point Crossover

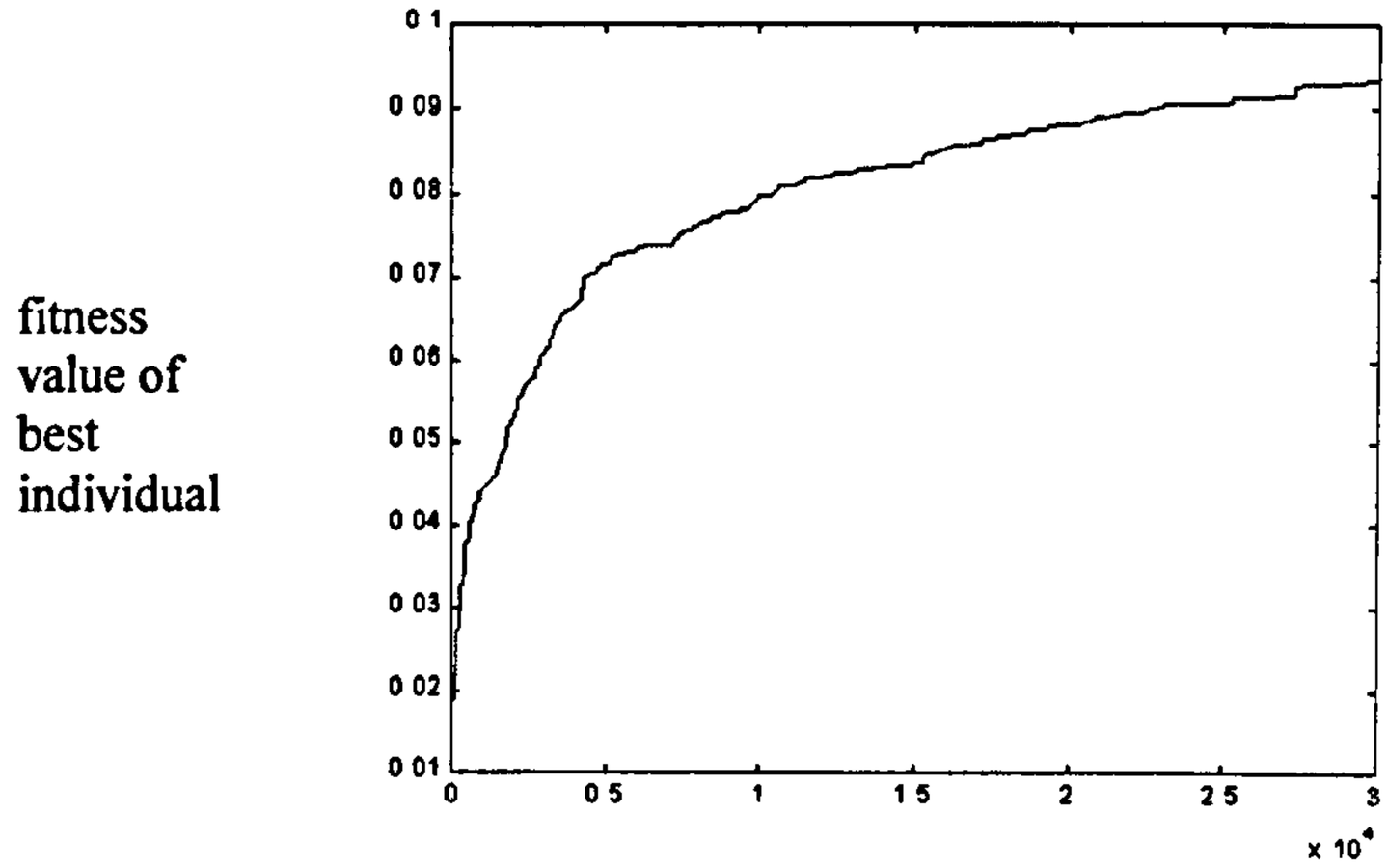
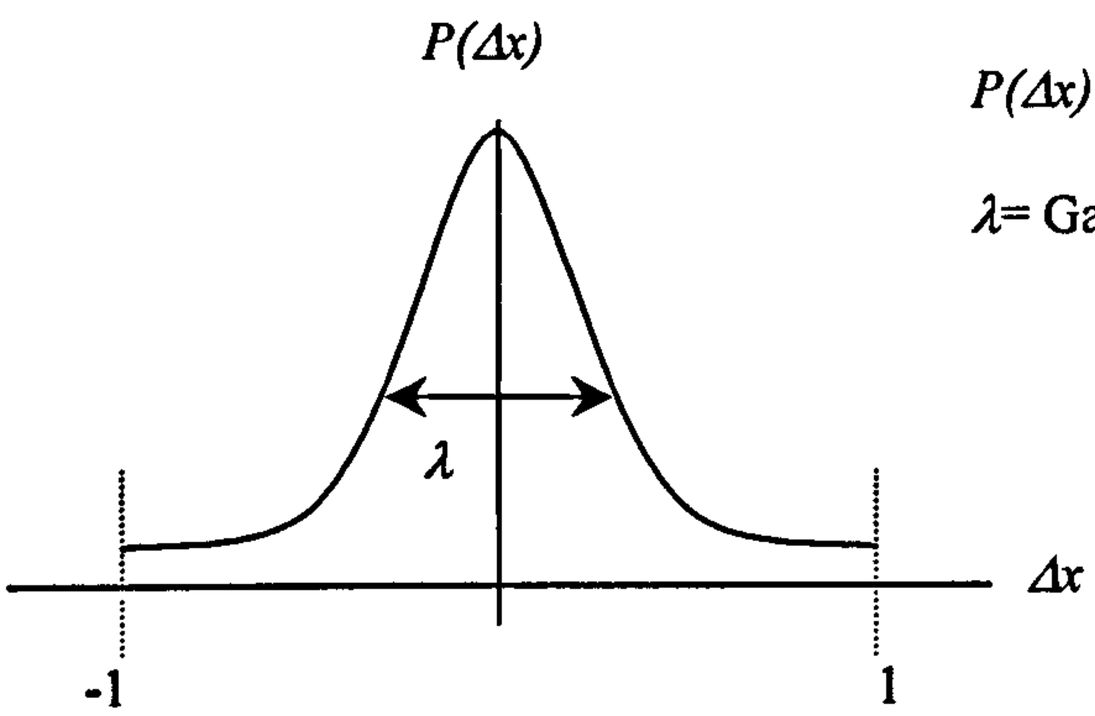


Figure 7.6.9 Three Point Crossover

7.6.2 Gaussian Mutation

Instead of uniform mutation a Gaussian function can be applied to the probability of the mutation value. At present any change in a gene's value by mutation has an equal probability of occurrence, so long as that change is within the limits set in the mutation operation. If a Gaussian function is applied about zero then smaller changes are more likely to occur than larger ones. It may be more useful to create large changes at the start of the evolution. However when the GA begins to struggle to find fitter chromosomes smaller changes may be better suited. A large change to an already good individual will very likely make it worse, good genome that need only to be fine tuned should be treated less violently. This can be done by reducing the width of the Gaussian curve λ as the GA progresses. The mutation function can even begin as uniform.

Figure 7.6.10 Gaussian Curve



Δx= change in gene value

$P(\Delta x)$=probabilty of change

λ= Gaussian curve width

Simulation comparisons are as follows:

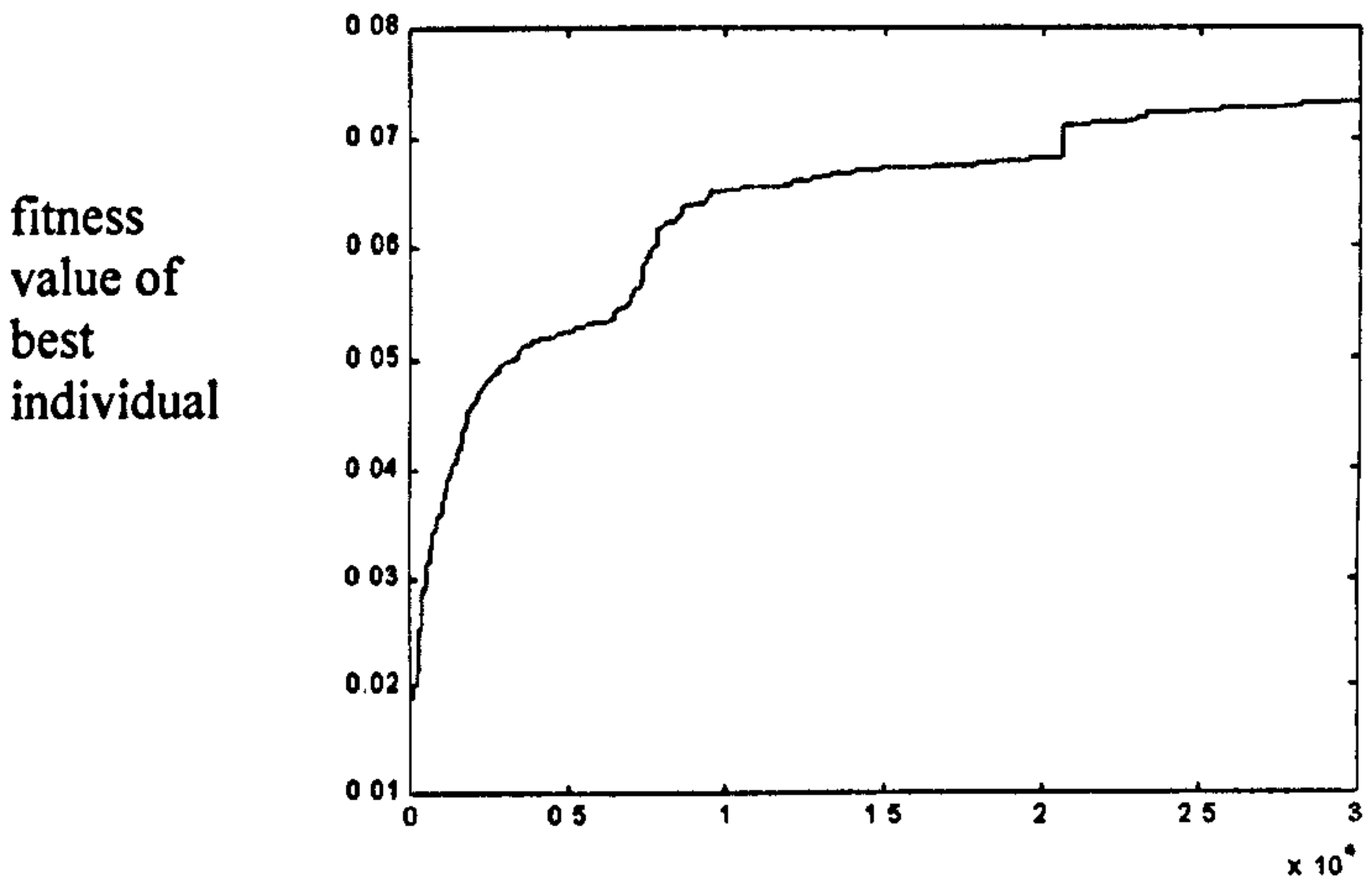


Figure 7.6.11 Uniform Mutation

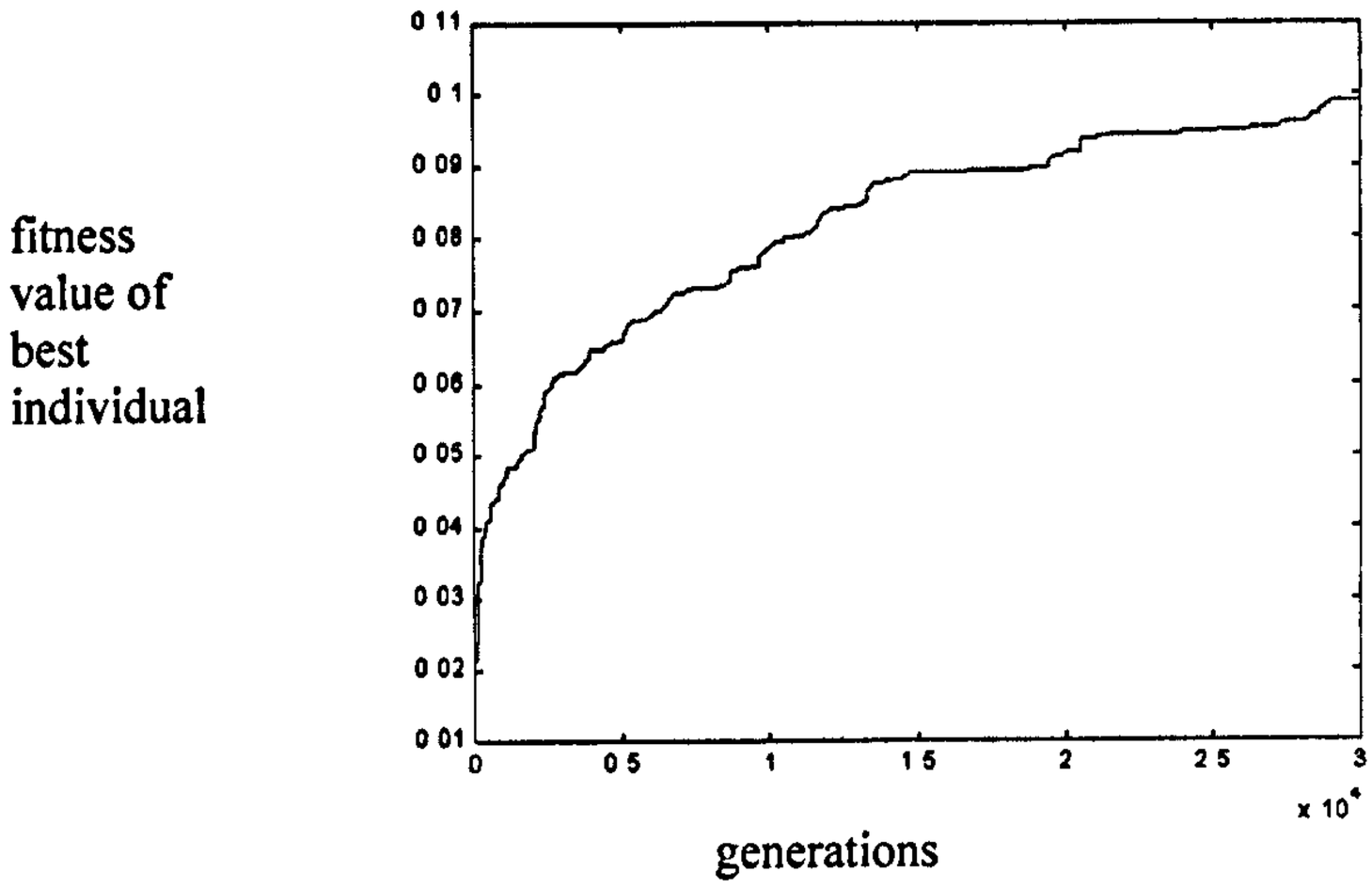


Figure 7.6.12 Gaussian Mutation
$\lambda = 1$

CHAPTER 8

Conclusions

8.1 Mammogram Asymmetry Analysis

The methods for region of interest selection (cropping and segmentation) described in section 5.2.1 are fast and computationally inexpensive compared to region growing. Better segmentations are also achieved. Calculation of the breast tissue relative area using integration leads to confirmation of similar findings by *Scutt* [82][83], the relationship between asymmetry factors and eventual cancer development. The non-scaling measure of asymmetry described in section 5.3.1 produced only cancer developing cases with more than a difference of 11,000.

Pattern feature extraction method using high pass filtering and the QTD algorithm (section 5.4) although highly sensitive and accurate is slow and computationally expensive. This asymmetry feature is not used in further processing by neural networks.

Further investigation into asymmetry analysis can include the use of Fractional Dimensions [72] as a measure difference in of 'roughness' of the grey scale image considered in three dimensions, although for images of such a high pixel number this can be computationally expensive.

8.2 Multi Layer Perceptron Development

Regular twelve fold validation in section 6.3.1 demonstrates the MLP did not learn. Figure 6.3.15 gives a large overlap of the validation outputs of the two classes and is echoed by the Receiver Operator Curve in figure 6.3.16 which is has no significant curvature with respect to the diagonal. The log likelihood error value during training, in both training and validation sets oscillates with a gradual decrease in oscillation amplitude over number of epochs and no consistent convergence is obtained.

In the case of the MLP using altered ordinal inputs described in section 6.2.1 the histogram of the validation outputs have less overlap (fig. 6.3.29) with respect to the outputs obtained from the MLP with no alteration in input. Again the Receiver Operator Curve (fig.6.3.30) still did not show any significant deviation from the linear diagonal however does deviate more than the ROC obtained from the MLP with no input alteration. This is quantified by the calculation of the area under the ROC curve shown in table 8.1.1.

The addition of missing data inputs corresponding to family history, thus increasing the training set to 852 patterns, increased the output overlap. This lead to a reduced deviation from the ROC diagonal in comparison to both regular MLP twelve fold cross validation with and without altered ordinal inputs.

Area Under ROC for Regular MLP	Area Under ROC for MLP with altered ordinal inputs	Area Under ROC for MLP with missing data
0.5614	0.5908	0.5198

Table 8.1.1 Areas under ROC curves.

Multi Layer Perceptron with these configurations and data set does not prove to be an adequate classifier. The alteration of the binary ordinal inputs into a distribution based real number scheme indicates that further pre-processing may produce improved results. The possibility of pre processing of categorised binary inputs with knowledge of similarities between the categories can be introduced in further work.

8.3 Conic Section Function Network Development

The training performance for the CSF network with this configuration and data set proved to be non convergent in the majority of validation sets in the twelve fold cross validation (figures 6.4.4 to 6.4.10). The initial LLE value for both validation and training set is value set by the OLS initiation procedure in figures 6.4.2 to 6.4.13. The EBP phase created an increase in LLE which did not converge in most instances. For this data set using CSF networks with this configuration are not trainable. The area under the ROC curve for the CSF network is 0.5746.

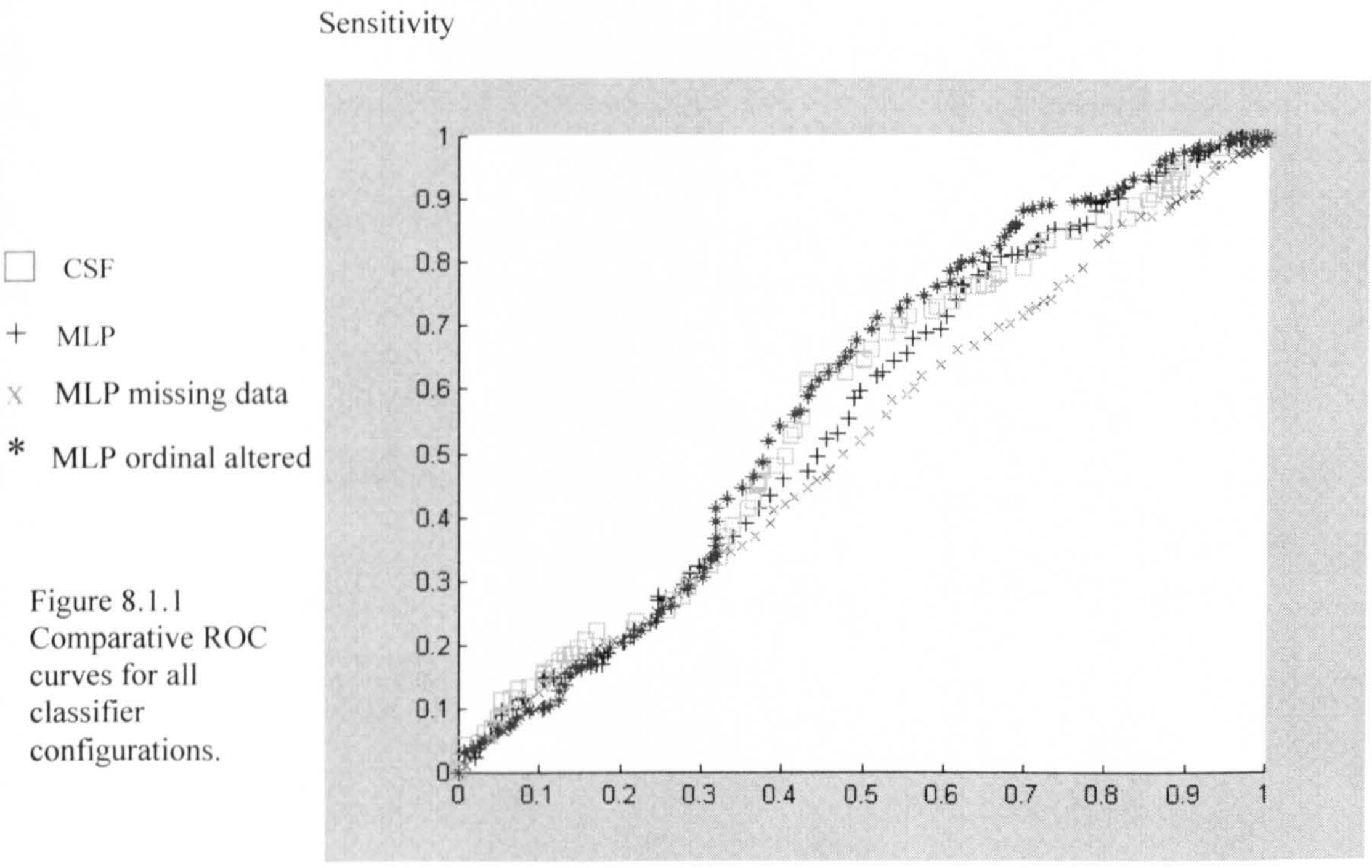


Figure 8.1.1 Comparative ROC curves for all classifier configurations.

8.4 Genetic Algorithm

The Algorithm shown in figure 7.6.4 based on the successful convergence of the second test algorithm (figure 7.6.3) encodes CSF network parameters can is tested. Figure 7.6.5 shows convergence in the desired direction however the fitness function of this trial does not reach realistic SSE values for satisfactory training.

Variations such as multiple point crossover and non uniform mutation does not improve the performance of the algorithm. In summary the best performing operators are elitist selection, uniform mutation and single point random crossover. This evolutionary method is found to be very slow and yielded poor performance.

Further investigation into this method can involve the adaptation of the genes from real to a set number of discrete values. This may have the effect increasing the pace of the algorithm by forcing the search to 'jump' to fixed points in the search space created by discrete genes.

References

[1] Ahmed E, '*Fractals and Chaos in Cancer Models*', International Journal of Theorectical Physics, 1993, Vol.32, No.2, pp.353-355.

[2] Amartur SC, Piraino D, Takefuji Y. *Optimization neural networks for the segmentation of magnetic resonance images.* IEEE Transactions on Medical Imaging; 11: 215-220. 1992.

[3] Andersson I, '*Radiographic Pattersn of the Mammary Parenchyma*'. Radiology. pp138: 59-62. 1981.

[4] Angeline P, Saunders G, Pollack J. '*An Evolutionary Algorithm that Constructs Recurrent Neural Networks*'. IEEE Transactions on Neural Networks vol. 5. no. 1. 54: 65. 1994.

[5] Astion ML and Wilding P, *Application of Neural Networks to the Interpretation of Laboratory Data in Cancer Diagnosis*, Clinical Chemistry, Vol. 38, No. 1, 1992.

[6] Baak JPA. *Manual of Quantitative Pathology in Cancer Diagnosis and Prognosis.* Springer-Verlag: New York, 1991

[7] Baeg S, Batman S, Dougherty ER, Kamat VG, Kehtarnavaz N, Kim S, Popov A, Sivakumar K, Shah R ,*'Unsupervised morphological granulometric texture segmentation of digital mammograms'*, Journal of Electronic Imagaing, 1999, Vol.8, No.1, pp.65-75

[8] BaumM, ChaplainMAJ, AndersonARA, DouekM, VaidyaJS, '*Does breast cancer exist in a state of chaos?*', European Journal Of cancer, 1999, Vol.35, No.6, pp.886-891.

[9] Billings S, Zheng G. *'Radial Basis Function Configuration Using Genetic Algorithms'.* Neural Networks vol. 8 no.6. 877-890. 1995.

[10] Bishop C. *'Neural Networks for Pattern Recognition'*. Oxford University Press. 1995

[11] Blatt N, Rubenstein J. '*The Canonical Coordinates Method for Pattern Recognition-II. Isomorphisms with Affine Transformations*'. Pattern Recognition Vol 27; No 1; 99-107. 1994.

[13] Boyd NF, O'Sullivan B, Campbell JE. *'Mammographic Signs risk factors for breast cancer'*. British Journal Cancer; 45. pp185-192. 1982.

[14] Burke HB. *Artificial neural networks for cancer research: outcome prediction*. Seminars in Surgical Oncology, 1994, 10:73--9.

[15] Byng JW, Yaffe MJ, Lockwood GA, Little LE, Tritchler DL, Boyd NF *'Automated analysis of mammographic densities and breast carcinoma risk'*, Cancer, 1997, Vol.80, No.1, pp.66-74.

[16] Carling A. *Introducing Neural Networks*. Sigma Press, Wilmslow UK. 1992.

[17] Chan HP, Lo SCB, Sahiner B, Lam KL, Helvie MA,*'Computer Aided Detection of Mammographic microcalcifications - Pattern-Recognition with an Artificial Neural-Network'* Medical Physics, 1995, Vol.22, No.10, pp.1555-1567.

[18] Chan HP, Sahiner B, Petrick N, Helvie MA, Lam KL, Adler DD, Goodsitt MM *'Computerized classification of malignant and benign microcalcifications on mammograms: Texture analysis using an artificial neural network'*, Physics in Medicine and Biology, 1997, Vol.42, No.3, pp.549-567

[19] Chen CH, Lee GG ,*'On digital mammogram segmentation and microcalcification detection using multiresolution wavelet analysis'*, Graphical Models And Image processing, 1997, Vol.59, No.5, pp.349-364.

[20] Chen CH, Lee GG *'Image segmentation using multiresolution wavelet analysis and expectation-maximization (EM) algorithm for digital mammography'*. International Jounral of Imaging Systems and Technology, 1997, Vol.8.

[21] Chen S, Cowan CFN, Grant PM, *'Orthognal Least Sqaures Leanring Algorithm for radial basis function networks'* , IEEE Transactions on neural networks, vol 2, no 2, 302-309, March 1991.

[22] Chen S, Wu Y, Alkadhimi K. *'A Two Layer Learning Method for Radial Basis Function Networks Using Combined Genetic and Regularised OLS Algorithms'*. Genetic Algorithms in Engineering Systems: Innovations and Applications. 12-14. 245:249. 1995.

[23] Cheng HD, Lui YM, Freimanis RI, '*A novel approach to microcalcification detection using fuzzy logic* ' IEEE Transactions on Medical Imaging, 1998, Vol.17, No.3, pp.442-450.

[24] Cohen P '*Fractal Cancer*' New Scientist, 1998, Vol.157, No.2118, p.14

[25] Dorfner G. '*Unified Framework For MLPs and RBFNs: Introducing Conic Section Function Networks*'. Cybernetics and Systems: An International Journal, 25. 511:554. 1994.

[26] Dorfnerr G, Porenta G. '*On Using Feed Forward Neural Networks for Clinical Diagnostic Tasks*'. Artifical Intelligence in Medicine. 1994.

[27] Eddy DM. *Screening for breast cancer*. Ann Intern Med. 111, pp389-399. 1989.

[28] Edstrom LE, Robson MC, Wright JK. '*A Method for the Evaluation of Minor Degrees of Breast Asymmetry. Plastic Reconstructive Surgey*'; 60. pp812-814. 1977.

[29] Egan J, '*Signal Detection Theory and ROC Analysis*'. New York: Academic, 1975.

[30] Ernster VL. Sacks ST, Peterson CA, Schweitzer RJ. '*Mammographic Parenchymal Patterns and Risk of Breast Cancer*'. Radiology; 134. 617-620. 1980.

[31] Eshelman L., Schaffer D. *'Real-Coded Genetic Algorithms and Interval Schemata'*. North American Philips Corporation, Genetic Algorithms. 187:203. 1992.

[32] Fisher RA. '*The use of Multiple Measurments in Taxonomic Problems. Annual Eugenics*'. 7. Part 2. 179-188. 1936.

[33] Gama J and Brazdil P. *Characterization of classification algorithms, In C.*, Progress in Artificial Intelligence, pages 189--200. Springer-Verlag, 1995.

[34] Gibson GJ, Cowan CFN. *On the decision regions of Multi layer Perceptrons*. Proc. IEEE Transactions on Neural Networks. Vol.3 No 4. 621-624. 1992.

[35] Goldberg D, *'Genetic Algorithms in Search, Optimisation and Machine Learning'*. Addison Wesley. 1989.

[36] Gonzalez RC, Woods RE. '*Digital Image Processing*'. Addison Wesley 1993.

[37] Goodwin PJ. Boyd NF. '*Mammographic Parenchymal Pattern and Breast Cancer Risk: A Critical Appraisal of the Evidence*'. American Journal of Epidemiology; 127. 1097-1108. 1988.

[38] Hejimans HJAM, Tuzikov AV. *'Similarity and Symmetry Measures of Convex Shapes Using Minkowski Addition'*. IEEE Trans. Pattern Analysis and Machine Intelligence. Vol 20; No 9; 980-993. 1998

[39] Helmrich SP, Shapiro S, Rosenberg L. '*Risk Factors for Breast Cancer*'. American Journal of Epidemiology. 117. 35-45. 1983.

[40] Huo ZM, Giger ML, Metz CE *'Effect of dominant features on neural network performance in the classification of mammographic lesions'* Physics in Medicine and Biology, 1999, Vol.44, No.10, pp.2579-2595

[41] Kalman BL, Reinus WR, Kwasny SC, Laine A, Kotner L *'Prescreening entire mammograms for masses with artificial neural networks: Preliminary results'* Academic Radiology, 1997, Vol.4, No.6, pp.405-414.

[42] Kim JK, Park HW, '*Statistical textural features for detection of microcalcifications in digitized mammograms* ' IEEE Transactions on Medical Imaging, 1999, Vol.18, No.3, pp.231-238

[43] Kim JK, Park JM, Song KS, Park HW '*Detection of clustered microcalcifications on mammograms using surrounding region dependence method and artificial neural network'* Journal of VLSI Signal Processing Systems for Signal Image and Video Technology, 1998, Vol.18, No.3, pp.251-262

[44] Kupinksi M, Anastasio M. '*Multiobjective Genetic Optimization of Diagnostic Classifiers with Implications for Generating Receiver Operating characteristic curves*'. IEEE Transactions On Medical Imaging, Vol. 18, No. 8, August 1999.

[45] Lado MJ, Tahoces PG, Mendez AJ, Souto M, Vidal JJ, *'A wavelet-based algorithm for detecting clustered microcalcifications in digital mammogram'* Medical Physics, 1999, Vol.26, No.7, pp.1294-1305

[46] Lefebvre F, Benali H, Gilles R, Kahn E, Di Paola R . *'A Fractal Approach To The Segmentaion Of Microcalcificaions In Didital Mammograms'*. Medical Physics, Vol.22, No.4, pp.381-390, 1995.

[47] Leinster SJ, *'The relationship between risk factors for Breast cancer and the Mammographic Parenchymal Pattern'* Thesis MD, University of Liverpool. 1989.

[48]Leondes, CT. *'Image Processing and Pattern Recognition (Neural Network Techniques and Applications)* Academic Press. 1998.

[49] Lisboa PJG, *'A review of evidence of health benefit from artificial neural networks in medical intervention'*. Neural Networks. 15. 11-39. 2002.

[50] Lisboa PJG, Vellido A, Aung MSH, El-Deredy W, Lee YYB, Kirkby SPJ. *'Quantification of uncertainty in tissue characterization with NMR spectra'*. 6th Scientific Meeting of the International Society for Magnetic Resonance in Medicine. Sydney, Australia. Abs No. 1851, April 1998.

[51] Lo JY, Baker JA, Kornguth PJ, Iglehart JD, Floyd CE Jr. *'Predicting breast cancer invasion with artificial neural networks on the basis of mammographic features.'*. Radiology. Apr;203(1):159-63.1997.

[52] Lo SCB, Lin JSJ, Freedman MT, Mun SK' *Application of artificial neural networks to medical image pattern recognition: Detection of clustered microcalcifications on mammograms and lung cancer on chest radiographs'* Journal of VLSI Signal Processing Systems for Signal Image and Video Technology, 1998, Vol.18, No.3, pp.263-274.

[53] Maniezzo V. *'Genetic Evolution of the Topology and Weight Distribution of neural Networks'*. IEEE Transactions on Neural Networks vol. 5. no. 1. 39: 53. 1994.

[54] Manning JT, Chamberlin AT. *'Fluctuating Asymmetry, Sexual Selection and Canine Teeth in Primates'*. Proc R Soc London B.; 251: 83-87. 1993.

[55] Manning JT, Scutt D, Whitehouse GH, Leinster SJ, Walton JW. *'Asymmetry and the Menstrual Cycle'*. Ethol Sociobiol; 17: 129-143. 1996.

[56] Manning JT, Scutt D, Whitehouse GH, Leinster SJ, Walton JW. *'Breast Asymmetry and Phenotypic Quality in Women'*. Evolution Human Behav; 18:1-13. 1997.

[57] Marsland JS Yildirim T, *' Optimisation by Back Propagation of Error in Conic Section Functions'* Conference on Vision, Recogntion Action: Neural Models of Mind and Machine, Boston May 28-31, 1997.

[58] Marsland JS' Yildirim T *'Improved Back Propagation Training Algorithm Using Conic Section Functions'*, IEEE International Conference on Neural Networks (ICNN'97) Houston Texas USA, June 9-12, 1997.

[59] Mattfeldt_T *'Nonlinear deterministic analysis of tissue texture: A stereological study on mastopathic and mammary cancer tissue using chaos theory'* Journal of Microscpoy-Oxford, 1997, Vol.185, No.Pt1, pp.47-66

[60] Metz BE, "*Basic principles of ROC analysis*," Seminars Nucl. Med., vol. VIII, no. 4, pp. 283--298, 1978.

[61] Michalewicz Z. *'Genetic Algorithms + Data Structures = Evolution Programs'*. Springer- Verlag. 1992.

[62] Minsky, M. and Papert, S. *'Perceptrons'*. MIT Press, Cambridge, MA. 1969.

[63] Møller AP. *'Female Preferences for apparently Symmetrical Male Sexual Ornaments in the Barn Swallow, Hirundo Rustica'*. Behave. Ecol and Sociobiol; 32: 371-376. 1993.

[64] Møller AP. *'Morphology and Sexual Selection in the Barn Swallow, Hirundo Rustica,Chernobyl, Ukriaine'*. Proc R Soc London B; 252: 51-57. 1993.

[65] Møller AP, Hogland J. *'Patterns of Fluctucating Asymmetry in Avain Feather Ornaments: Implications for Models of Sexual Selection'*. Proc R Soc London B., 245: 1-5. 1991.

[66] Møller AP, Soler M, Thornhill R. *'Breast Asymmetry, Sexual Selection and Human Reproductive Success'*. Ethol and Sociobiol; 16; 207-219. 199

[67] Moody J, Darken C, *'Learning with Localised Receptive Fields'*. Proc of the 1988 Connectionist Summer School 133-143

[68] Moody J, darken C ,' *Fast Learning in network of locally tuned processing units* . Neural Computation, 1:281-294, 1989.

[69] McCulloch, W. S. and Pitts, W. H. *'A logical calculus of the ideas immanent in nervous activity'*. Bulletin of Mathematical Biophysics, *5*:115-133. 1943.

[70] Nagel RH, Nishikawa RM, Papaioannou J, Doi K ,*'Analysis of methods for reducing false positives in the automated detection of clustered microcalcifications in mammograms'* Medical Physics, 1998, Vol.25, No.8, pp.1502-1506

[71] Palmer AR, Strobeck C, *'Fluctuating Asymmetry: Measurement analysis and Patterns'*. Annual Review Ecolological Systems; 17. 391-421. 1986.

[72] Peleg S, Naor J, Hartley R, Avnir D. *'Mutiple Resolution Texture Analysis and Classification'*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-6. No. 4. 518-523. 1984.

[73] Ravdin PM; Clark GM. *'A practical application of neural network analysis for predicting outcome of individual breast cancer patient's*. Breast Cancer Research and Treatment, 1992, 22(3):285--93.

[74] Rew_DA *'Tumour biology, chaos and non-linear dynamics'* European Journal Of Surgical Oncology, 1999, Vol.25, No.1, pp.86-89

[75] Ripley BD. *'Can statistical theory help us use neural networks better*?' 29th Symposium on the interface: Computer Science and Statistics. 1997

[76] Rivlin E, Weiss I. *'Local Invariants for Recognition'*. IEEE Trans. Pattern Analysis and Machine Intelligence. Vol 17; No 3; pp226-238. 1995.

[77] Roberts SJ. *'Extreme value statistics for novelty detection in biomedical signal processing. Advances in Medical Signal and Information Processing'*. Conference Publication No 476. IEEE. 166-171. 2000.

[78] Rumelhart D, Hinton G, Williams R, *'Learning internal representations by error propagation. In Paralell distributed Processing* , Vol 1 Cambridge MA: MIT Press, pp318-362 1987.

[79] Safftlas AF, Szklo M. *Mammographic Parenchymal Patterns and Breast Cancer Risk.* Epidimiology Review; 9, pp146-174. 1987.

[80] Schenone A, L. Andreucci, V. Sanguinetti, and P. Morasso. *'Neural Networks for prognosis in breast cancer'*. Physica Medica, IX(Supplement 1):175--178, June 1993.

[81] Schmidt F, Sorantin E, Szepesvari C, Graif E, Becker M, Mayer H, Hartwagner_K *An automatic method for the identification and interpretation of clustered microcalcifications in mammograms.*,Physicis in Medince and Biology, 1999, Vol.44, No.5, pp.1231-1243

[82] Scutt D, Manning JT, Whitehouse GH, Leinster SJ, Massey CP. *'The Relationshipbetween Breast Asymmetry, Breast Size and the occurence of Breast Cancer'.* British Journal of Radiology; 70: 1017-1021. 1997.

[83] Scutt D, *'Fluctuating and cyclical asymmetry: their relationship to breast cancer and fertility'* PhD Thesis , The University of Liverpool, May 1998

[84] Sedivy R, Mader RM *Fractals, chaos, and cancer: Do they coincide?* Cancer Investigation, 1997, Vol.15, No.6, pp.601-607

[85] Stefanoyiannis AP, Costaridou L, Sakellaropoulos P, Panayiotakis G. *'A digital Density Equalisation Technique to Improve Visualisation of Breast Periphery in Mammography'.* British Jounral of Radiology; 73; 410-420. 2000

[86] Stone M. *'Cross-validatory choice and assessment of statistical predictions'* (with discussion). Journal of the Royal Statistical Society B, 36:111-147, 1974.

[87] Tang K, Chan C, Man K, Kwong S. *' Genetic Structure for NN Topology and Weights Optimization'*. Genetic Algorithms in Engineering Systems: Innovations and Applications. 12-14. 250:255. 1995.

[88] Toniolo PG , Kato I, Beinart C, Bleich A, Su S, Kim M,. *'A Nested Case Control Study of Mammographic Patterns, Breast Volume and Breast Cance'r.* Cancer Causes and Control. New York City USA. 6. 431-438. 1995.

[89] Tsujii O, Freedman MT, Mun SK *Classification of Microcalcifications in digital mammograms using trend-oriented radial basis function neural network'* Pattern Recognition, 1999, Vol.32, No.5, pp.891-903

[90] Van Valen L. *A Study of Fluctuating Asymmetry.* Evolution. 16. 125-142. 1962.

[91] Vandenbussche F. *Asymmetries of the Breast: A Classification System.* Aesthetic Plastic Surgery. 8, 27-36. 1984.

[92] Veenland J, Grashuis J, Van der Meer F, Beckers A, Gelsema E. *'Estimation of Fractal Dimension in Radiographs',* Medical Physics. No 23. Vol 4. 585:594. 1996

[93] Whitehead B, Choate T. *'Co-operative Competitive Genetic Evolution of Radial Basis Function Centres and Widths for Time Series Prediction'.* IEEE Transactions on Neural Networks. vol. 7. no. 4. 869:880. 1996.

[94] Whitley D, Dominic S, Das R. *'Genetic Reinforcement Learning with Multi-layer Neural Networks'.* Proceedings from the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann. 101:108. 1991.

[95] Wilding P, M.A. Morgan, A.E. Grygotis, M.A. Sho#ner, and E.F. Rosato. *Application of backpropagation neural networks to diagnosis of breast and ovarian cancer.* Cancer Letter, 77:145--153, 1994.

[96] Winsberg F, Elkin M, Macy J, Bordaz V, Weymouth W. '*Detection of radiographic abnormalities in mammograms by means of optical scanning and computer analysis.*' Radiology ;89:211-215.1967.

[97] Wolfe JN. *Breast Patterns as an Index of Risk for Developing Breast Cancer.* American Journal of Radiology. 126. 1130-1139. 1976.

Woods K and K. W. Bowyer, '*Generating ROC curves for artificial neural networks,*' IEEE Trans. Med. Imag., vol. 16, pp. 329--337, June 1997.

[98] Wu YZ, Doi KN, Giger ML, Nishikawa RM '*Computerized Detection of Clustered Microcalcifications In Digital Mammograms -Applications of Artificial Neural Networks* Medical Physics 1992, Vol.19, No.3, pp.555-560

[99] Yaffe MJ, Boyd NF, Byng JW, Jong RA, Fishell E, Lockwood GA, Little LE, Tritchler DL *'Breast cancer risk and measured mammographic density'* European Journal of Cancer Prevention, 1998, Vol.7, No.S1, pp.S47-S55

[100] Yao X, Liu Y. ' *A New Evolutionary System for Evolving Artificial Neural Networks* '.IEEE Transactions on Neural Networks. vol. 8. no. 3. 694:713. 1997.

[101] Yildirim T, Ozyilmaz L. *'Dimensionality reduction in conic section function neural network. Sadhana.'* Vol 27. part 6. 675-683. 2002

[102] Zhang W, DoiK, Giger ML, Nishikawa RM, Schmidt RA' *An improved shift-invariant artificial neural network for computerized detection of clustered microcalcifications in digital mammogram's* Medical Physics 1996, Vol.23, No.4, pp.595-601

[103] Zhang W, Yoshida H, Nishikawa RM, Doi K '*Optimally weighted wavelet transform based on supervised training fordetection of microcalcifications in digital mammograms*',Medical Physics, 1998, Vol.25, No.6, pp.949-956

[104] Zheng BY, Qian W, Clarke LP ',*Digital mammography: Mixed feature neural network with spectral entropy decision for detection of microcalcifications*. IEEE transactions on medical imaging, 1996, Vol.15, No.5, pp.589-597

Appendix A

Asymmetry Analysis of Digitised Mammogram Pairs and the Relationship to Cancer Predisposition.

Aung M S H*, Scutt D+, Marsland J S*.

* Department of Electrical Engineering and Electronics, University of Liverpool

+Department of Medical Imaging, University of Liverpool

Keywords: Asymmetry Analysis, Breast Cancer, Medical Imaging

Contact Information

Dr J S Marsland,

Dept of Electrical Engineering and Electronics,

University of Liverpool,

Brownlow Hill, Liverpool, L69 3GJ, UK.

Tel: +44 151 794 4536

Email: marsland@liv.ac.uk

Abstract

Asymmetry is a well-known phenotypic indicator for developmental stability. This paper demonstrates methods for measuring asymmetry of breasts using digitised mammograms. Although diagnosis of cancer was not given to any of the mammograms used at the time of mammography, eventual development did occur in half of the set. The relationship between cancer and volume asymmetry has been demonstrated previously. This paper follows on to introduce accurate area calculation and shape comparison methods.

Introduction

Humans, in common with most other vertebrates, show bilateral symmetry in paired morphological traits such as ear size, digit length and breast volume. Perfect symmetry may be disturbed by a number of intrinsic and extrinsic factors including the secretion of hormones such as oestrogen [5][6]. The small deviations from perfect symmetry which result from such factors are termed fluctuating asymmetry (FA). FA is a well established biological measure of developmental stability, and is one of many issues at the interface between biology and medicine which offer valuable information at the whole organism level. Such comprehensive information is a concept familiar to, and frequently used by biologists, but often ignored in medicine.

FA tends to be greater in sexually selected traits, such as breasts than non-sexually selected characters [9][4]. The former are more liable to be disrupted during development because they are generally elaborate in design and are therefore highly susceptible to mutation [7][8]. *Møller et al* [11] found that large breasts had more FA than small breasts, breast FA was higher in nulliparous women, and that breast FA was a predictor of fecundity. The large between-individuals differences in size and asymmetry of breasts could be indicative of differences in developmental stability, and possibly disease predisposition. Breast volume FA, as measured from mammograms, is related to several of the known risk factors for breast cancer [5][6] and patients with breast cancer have higher breast FA than age-matched healthy women [12].

Methodologies

This was a retrospective study on women who had volunteered for mammography during the period 1978-1988 in the Liverpool study on breast cancer risk factors [3]. Some 12,900 women underwent mammography during this period, and those who were disease-free at that time but had subsequently gone on to develop breast cancer during the intervening 22 years to date were identified by a data-match exercise with records from the North West Cancer Registry.

Standard and non-standard image specific processing techniques [14] are employed for the extraction of features significant to FA. In *Scutt et al*, [12] difference in volume was evaluated for FA by way of parabolic approximation using the base length and maximum width of *cranio-caudal* view mammograms. This study focuses on area and shape differences also using *cranio-caudal* mammograms.

Part 1: Image segmentation.

Each image contains sections that are irrelevant to this study, such as background radiation, name tags, left right indicators and segments produced from the digitisation process. After the inspection of the images and its grey level cross sections, constant characteristics and parameters were found. Figure 2

original image

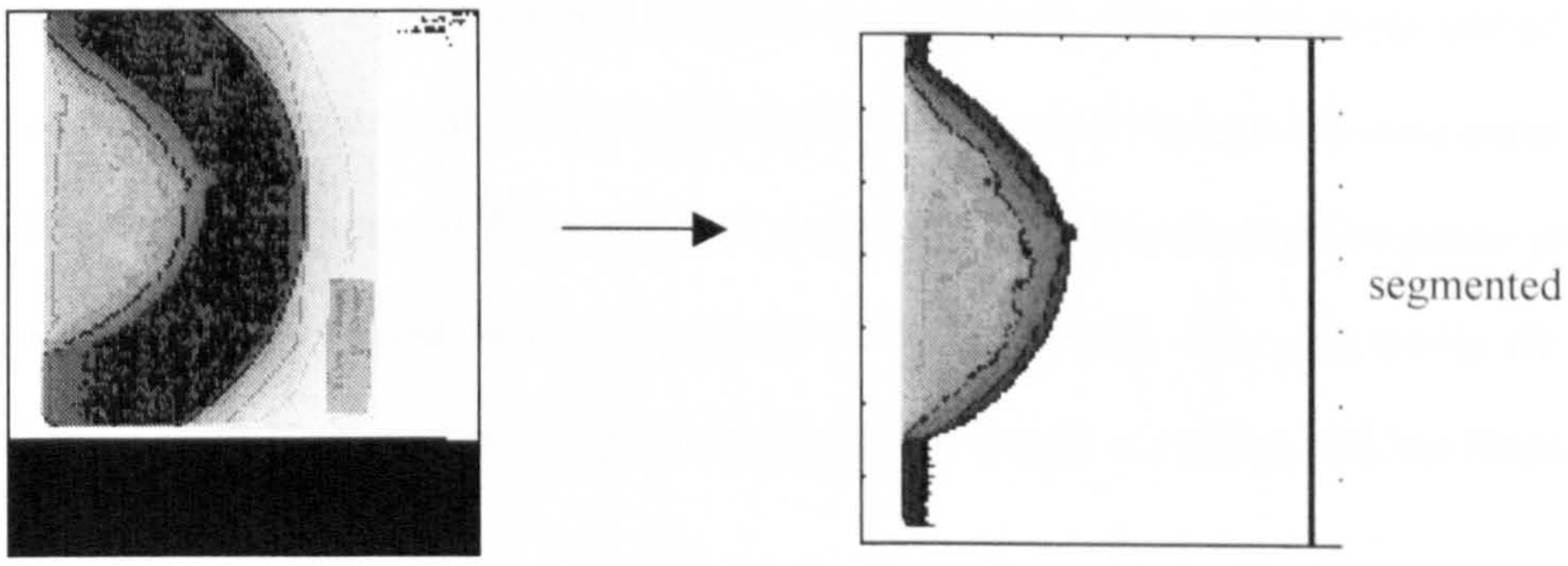


Fig 1

below shows the grey levels of a horizontal cross section of the image before segmentation in figure 1.

White is represented as 255 and black at 0.

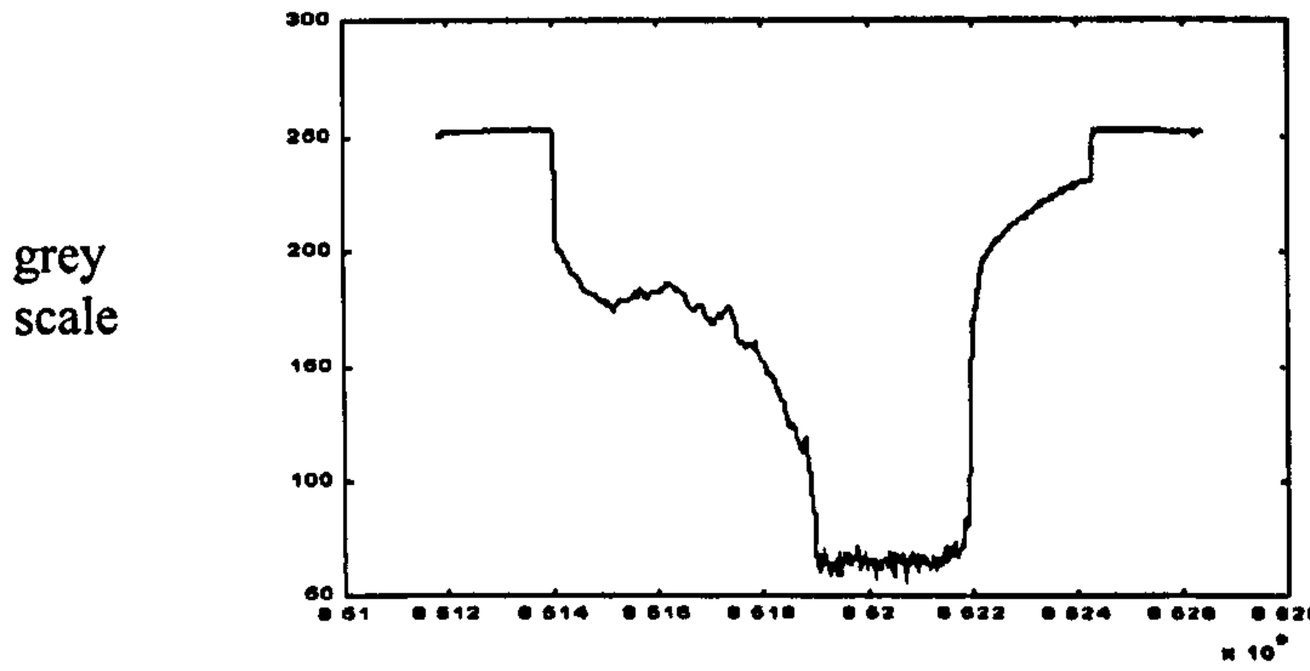


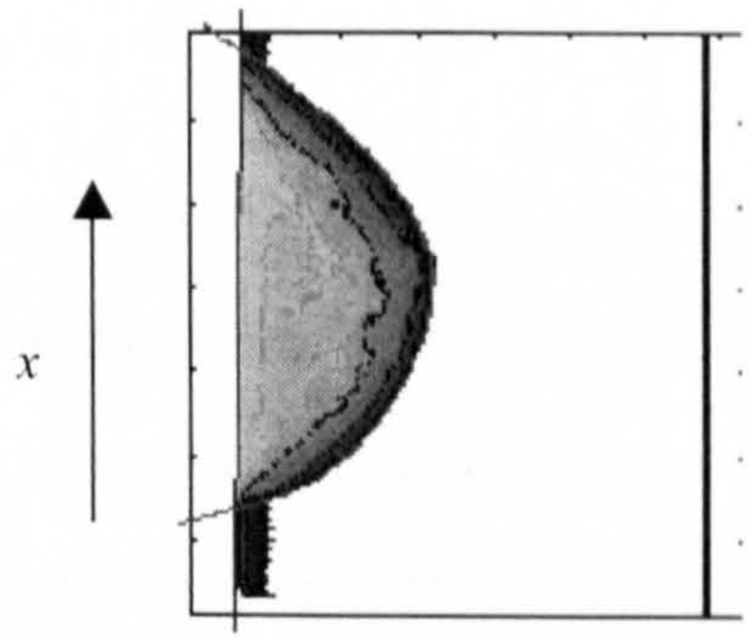
Fig 2

The characteristics of the cross section in fig.2 were found to be typical among the images and a rule-based algorithm specific to images with such cross sections was established. The algorithm identified tissue pixels from pixels in the image that are outside of the breast. From left to right the algorithm checked each pixel in each row of the image. Once the levels dropped to that of the minimum, which was always found to be that of the background radiation, the process halts and proceeds to the next row. This gave rise to segmented images of breast tissue as shown in Fig 1.

Part 2: Curve fitting and Area Calculation.

After segmentation the FA features can be extracted. Using the co-ordinates of the pixels at the curved edge of the segmented image a 6th order polynomial is best fitted to those pixels using a standard linear regression method. For the straight edge of the breast area, a straight line is fitted using two points at that edge. The area of the encompassed section is calculated by integration. The area under the polynomial (coefficients a to g) is calculated, using the two crossover points x'_1 and x'_2 with the fitted line as the limits (eqn 1).

Fig 3



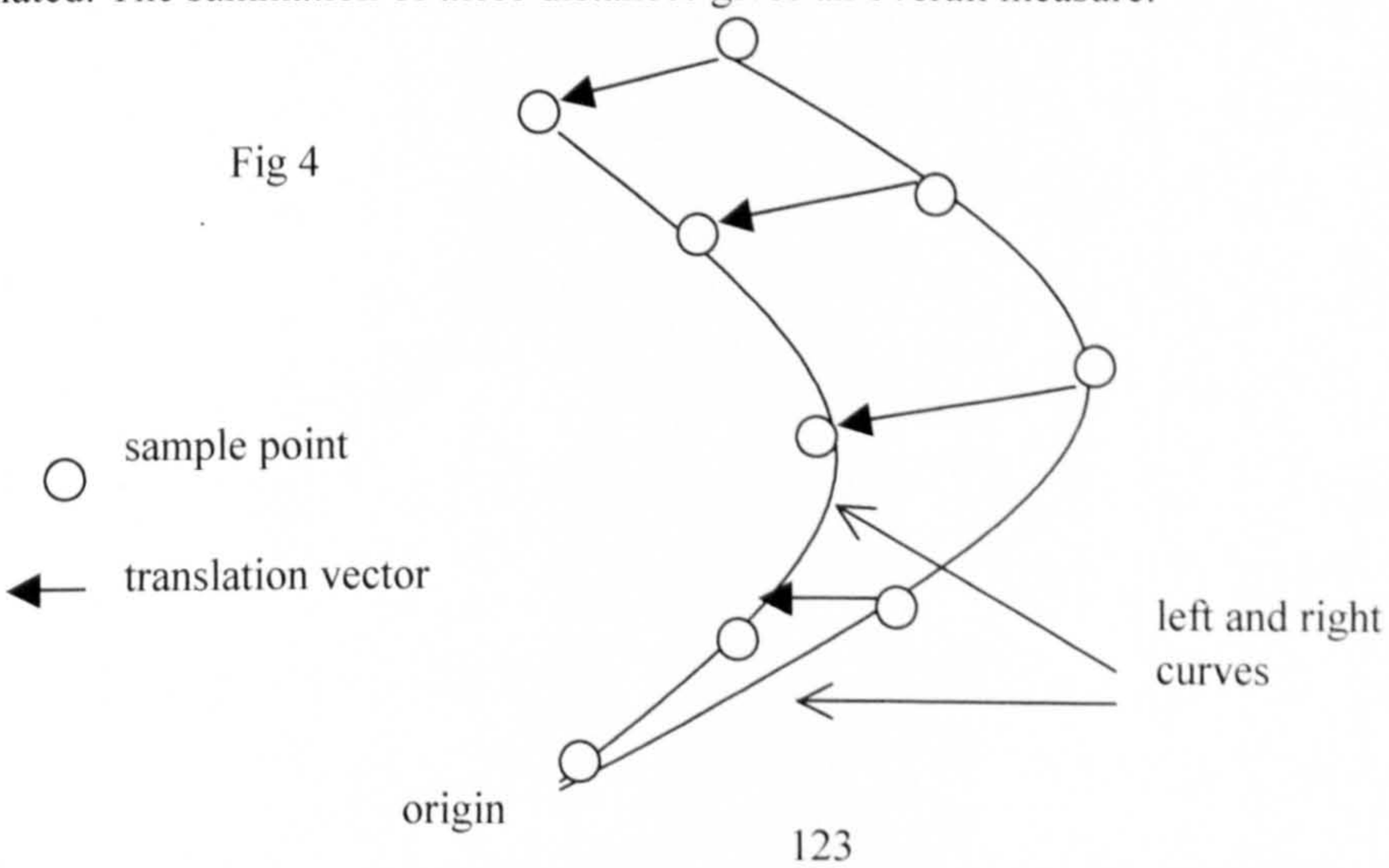
$$A=\left(\int_{x_1}^{x_2} ax^6+bx^5+cx^4+dx^3+ex^2+fx+g\right)-\left(\int_{x_1}^{x_2} mx+n\right) \qquad \text{eqn.1}$$

Relative area difference is then calculated from the area difference among pairs divided by the sum of the areas.

Part 3: Shape Similarity Measures.

Various complex measures for similarity between two shapes exist [2][1]. Measures obtained from isomorphic algorithms for two-dimensional shapes [2] are often used. The measure in this study is the morphological translation of 100 points from one polynomial to another. In the case of the images in this study the variant attribute is represented by the relative positions between the curved and straight edge of the segmented image. All of the images have crossover points (used as limits of integration in part 1). The vertical distance between these points is used to identify 100 equally spaced sample points (including the crossover points). This algorithm starts by mapping the lowest of these points to the origin (both curves). The Euclidean distance between the remaining points on both curves can then be calculated. The summation of these distances gives an overall measure.

Fig 4



However this is not a pure representation of shape as it is influenced by the curve size. Using the maximum widths, each curve is scaled (maximum equals 1) in the horizontal direction. Scaling in the vertical direction is achieved by simply discounting the vertical element of the translation vector thereby assuming that all sample points are at equal vertical intervals. The summation of the horizontal differences of the 100 points is then calculated.

Two measures are obtained:

1. Non-scaled shape differences between curve pairs. (size dependant).
2. Scaled shape differences between curve pairs. (size independent).

Results

The following histograms (Fig. 5, 6, 7) present the results for a test batch of approximately 200 pairs of mammograms. At the time of mammography no diagnosis of cancer had been made. Subsequently half of the sample developed breast cancer (labelled cancer on figures) and the rest have not in the intervening 22 years. The number of occurrences is plotted against intervals of ascending FA with the lower limit inclusive to the interval.

Relative Area Difference

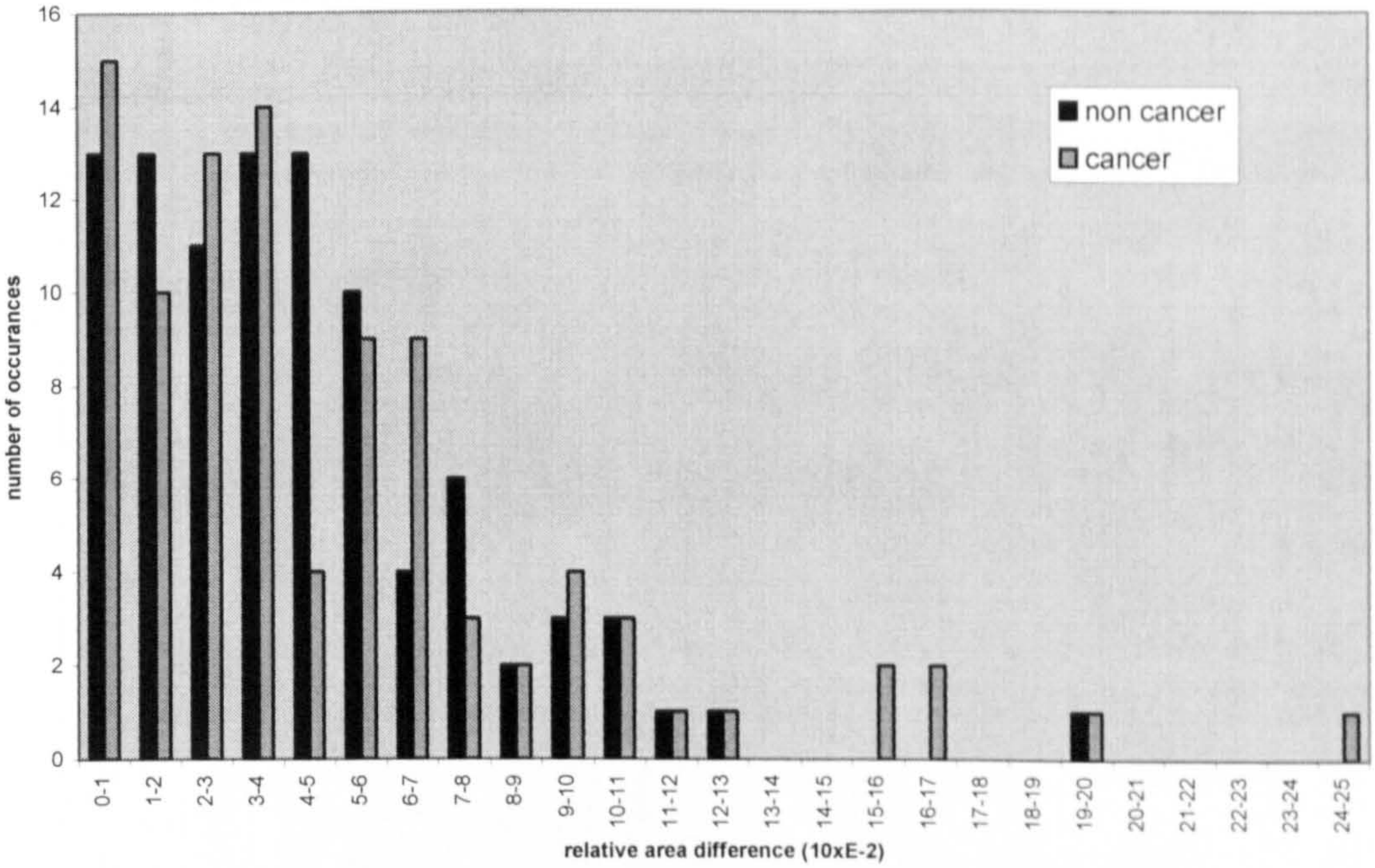


Fig 5

In accordance with well-understood principles on pre-dispositions [12], high FA is more prevalent in the cancer sets. A comparison of this prevalence between the three different measures of FA can be made. The most notable being the non scaled difference for shape similarity (Fig. 7). The contrast between the cancer and non-cancer sets for scaled difference (Fig. 6) is not as great as the features that are influenced by breast size (relative area and non scaled difference). High symmetry is also prevalent in all cases although a minimum level of asymmetry exists for the measures based on shape. This is not found in relative area difference.

Scaled Shape Difference

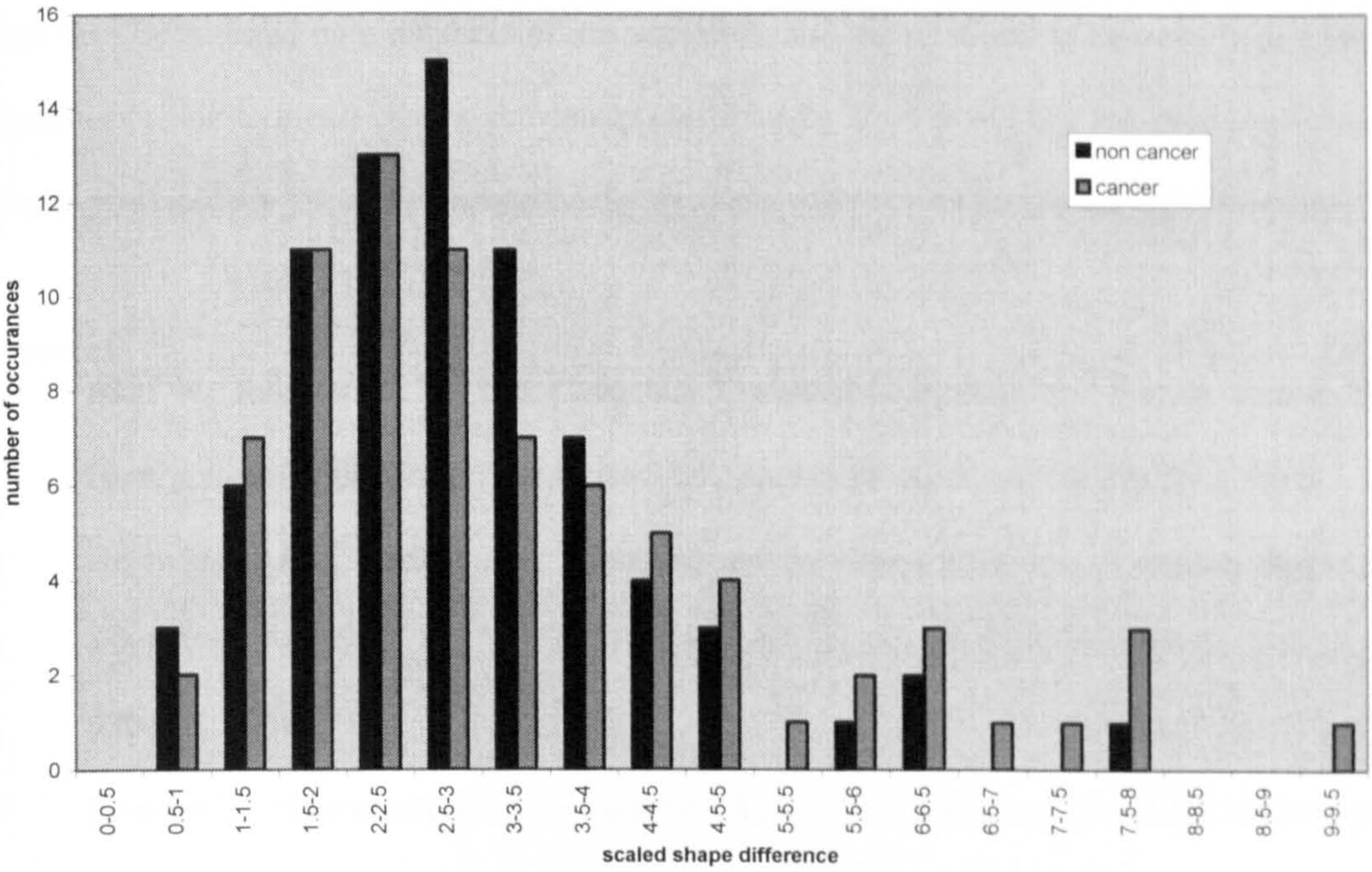


Fig 6

Non Scaled Shape Difference

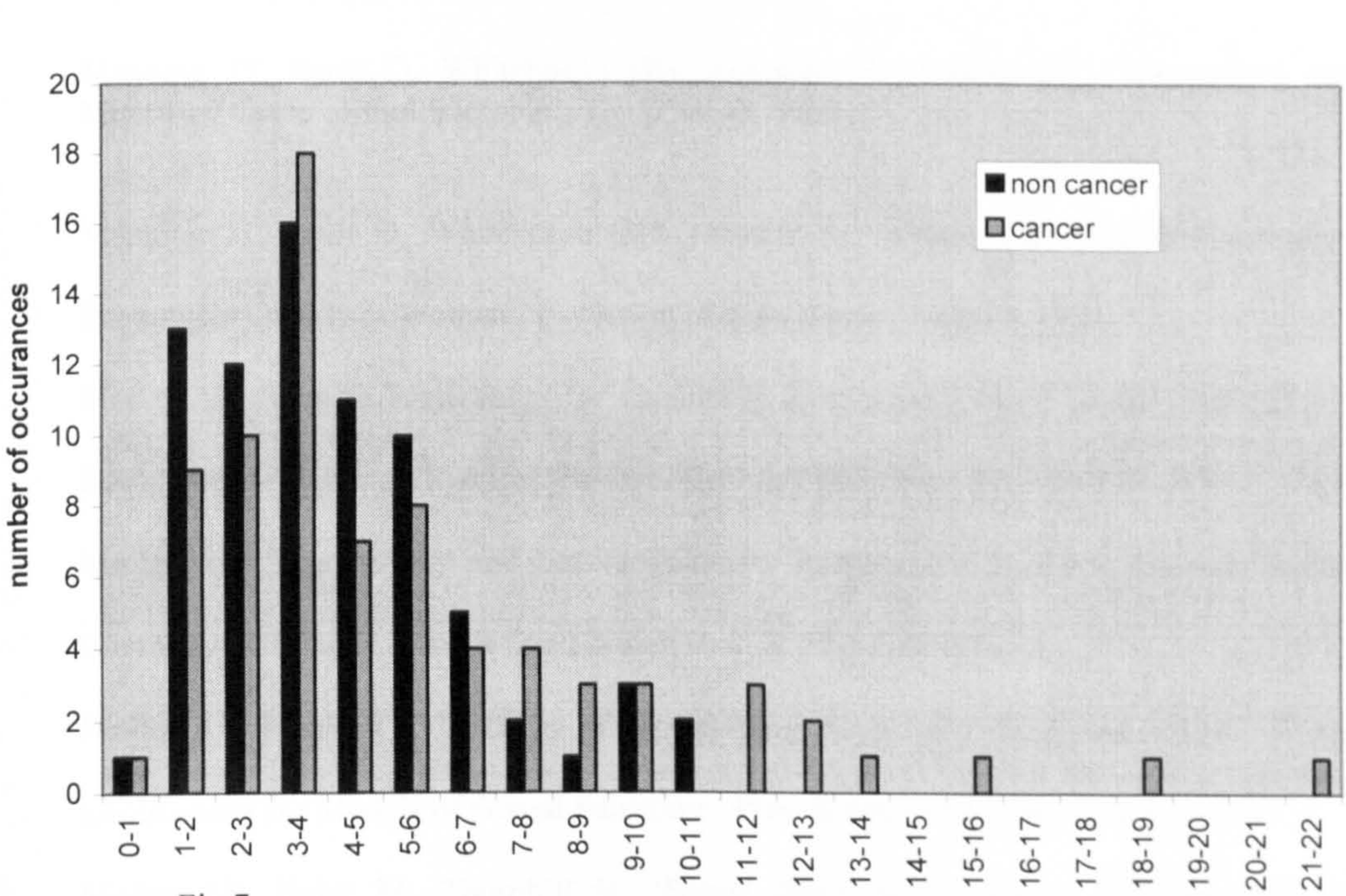


Fig 7

Scutt et al [12] demonstrated predisposition trends using difference in volume. The measures in the above figures show similar results and give grounds for such versions of FA to be investigated on a larger scale.

Conclusion

Three new measures of breast asymmetry have been developed using image processing techniques. These have been tested on a database of mammograms and the relationship between high asymmetry and predisposition to breast cancer, previously identified by *Scutt et al* [12], has been confirmed. The methods presented are fast and computationally inexpensive strategies for measuring asymmetry.

References

[1] Blatt N, Rubenstein J. 'The Canonical Coordinates Method for Pattern Recognition-II. Isomorphisms with Affine Transformations'. Pattern Recognition Vol 27; No 1; 99-107. 1994.

[2] Hejimans HJAM, Tuzikov AV. 'Similarity and Symmetry Measures of Convex Shapes Using Minkowski Addition'. IEEE Trans. Pattern Analysis and Machine Intelligence. Vol 20; No 9; 980-993. 1998.

[3] Leinster SJ, 'The relationship between risk factors for Breast cancer and the Mammographic Parenchymal Pattern' Thesis MD, University of Liverpool. 1989.

[4] Manning JT, Chamberlin AT. 'Fluctuating Asymmetry, Sexual Selection and Canine Teeth in Primates'. Proc R Soc London B.; 251: 83-87. 1993.

[5] Manning JT, Scutt D, Whitehouse GH, Leinster SJ, Walton JW. 'Asymmetry and the Menstrual Cycle'. Ethol Sociobiol; 17: 129-143. 1996.

[6] Manning JT, Scutt D, Whitehouse GH, Leinster SJ, Walton JW. 'Breast Asymmetry and Phenotypic Quality in Women'. Evolution Human Behav; 18:1-13. 1997.

[7] Møller AP. 'Female Preferences for apparently Symmetrical Male Sexual Ornaments in the Barn Swallow, Hirundo Rustica'. Behave. Ecol and Sociobiol; 32: 371-376. 1993.

[8] Møller AP. 'Morphology and Sexual Selection in the Barn Swallow, Hirundo Rustica, in Chernobyl, Ukriaine'. Proc R Soc London B; 252: 51-57. 1993.

[9] Møller AP, Hogland J. 'Patterns of Fluctucating Asymmetry in Avain Feather Ornaments: Implications for Models of Sexual Selection'. Proc R Soc London B., 245: 1-5. 1991.

[10] Møller AP, Soler M, Thornhill R. 'Breast Asymmetry, Sexual Selection and Human Reproductive Success'. Ethol and Sociobiol; 16; 207-219. 199

[11] Rivlin E, Weiss I. 'Local Invariants for Recognition'. IEEE Trans. Pattern Analysis and Machine Intelligence. Vol 17; No 3; 226-238. 1995.

[12] Scutt D, Manning JT, Whitehouse GH, Leinster SJ, Massey CP. 'The Relationship between Breast Asymmetry, Breast Size and the occurence of Breast Cancer'. British Journal of Radiology; 70: 1017-1021. 1997.

[13] Stefanoyiannis AP, Costaridou L, Sakellaropoulos P, Panayiotakis G. 'A digital Density Equalisation Technique to Improve Visualisation of Breast Periphery in Mammography'. British Jounral of Radiology; 73; 410-420. 2000.

[14] Gonzalez RC, Woods RE. 'Digital Image Processing'. Addison Wesley 1993.

Appendix B

MATLAB CODES

CREATE DATA CODE

```
targets=zeros(564,2);

for i=1:564
   if caorno(i)==1
      targets(i,1)=0.9;
      targets(i,2)=0.1;
   else
      targets(i,1)=0.1;
      targets(i,2)=0.9;
   end
end


%vol=[rvol lvol];

relvol=zeros(564,1);

for ii=1:564
   difftemp=(rvol(ii)-lvol(ii));
   sumtemp=(rvol(ii)+lvol(ii));
   relvol(ii)=(sqrt((difftemp)^2))/sumtemp;
end


fh=zeros(564,4);

for iii=1:564
   if famhis(iii)==0
      fh(iii,1)=1;
   else
      if famhis(iii)==1
         fh(iii,2)=1;
      else
         if famhis(iii)==2
            fh(iii,3)=1;
         else
            fh(iii,4)=1;
         end
      end
   end
end


lpch=zeros(564,4);

for iii=1:564
   if lparench(iii)==1
      lpch(iii,1)=1;
   else
      if lparench(iii)==2
         lpch(iii,2)=1;
      else
         if lparench(iii)==3
            lpch(iii,3)=1;
```

```
            else
               lpch(iii,4)=1;
            end
         end
      end
end


rpch=zeros(564,4);

for iii=1:564
   if rparench(iii)==1
      rpch(iii,1)=1;
   else
      if rparench(iii)==2
         rpch(iii,2)=1;
      else
         if rparench(iii)==3
            rpch(iii,3)=1;
         else
            rpch(iii,4)=1;
         end
      end
   end
end


agm=zeros(564,6);

for iii=1:564
   if ageatmen(iii)<=11
      agm(iii,1)=1;
   else
      if ageatmen(iii)==12
         agm(iii,2)=1;
      else
      if ageatmen(iii)==13
         agm(iii,3)=1;
      else
         if ageatmen(iii)==14
         agm(iii,4)=1;
         else
            if ageatmen(iii)==15
            agm(iii,5)=1;
            else
               agm(iii,6)=1;
            end
         end
      end
   end
end
```

```
%%%%%%% Find crossover points of curve and line for limits of
Intergration%%%%%%
limit=size(yyyy);
hlimit=limit/2;

lim=limit(2);

halflimitemp=hlimit(2);
halflim=round(halflimitemp);

for count=1:halflim
ydiffs=yyyy(count)-yy(count);
sqydiffs(count)=sqrt(ydiffs^2);
end


%IA=find(sqydiffs==min(sqydiffs));
IA=126;
for count=halflim:lim
ydiffs=yyyy(count)-yy(count);
sqydiffs(count)=sqrt(ydiffs^2);
end

sqydiffslatter=sqydiffs(halflim:lim);
IBtemp=find(sqydiffslatter==min(sqydiffslatter));

IB=IBtemp+(halflim-1);

%%%%%calculate area under curve%%%%%%%%%%%%%


syms intx;
curvearea=int(((curve(1)*intx^6)+(curve(2)*intx^5)+(curve(3)*intx^4)+
(curve(4)*intx^3)+(curve(5)*intx^2)+(curve(6)*intx)+(curve(7))),intx,
IA,IB);



%%%%%%calculate area under line%%%%%%%%%%%%%%%%%

syms intxx;

linearea=int((mm*intxx)+cc,intxx,IA,IB);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

breastarea=curvearea-linearea;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


disp('area=');
disp(breastarea);


disp('Coeff7=');
```

```
disp(curve(1));



disp('Coeff6=');
disp(curve(2));

disp('Coeff5=');
disp(curve(3));


disp('Coeff4=');
disp(curve(4));


disp('Coeff3=');
disp(curve(5));


disp('Coeff2=');
disp(curve(6));

disp('Coeff1=');
disp(curve(7));
```

Region Growing

```
% region growing algorithm

openadavies;
initi=700;
initj=700;
maxdiff=5;
 %figure;

  %axis([0 1460 0 1752]);
  %hold on;

t=1;
stack(t)=mamm(initi,initj);
stacksize=size(stack);
coordstack(t,1)=initi;
coordstack(t,2)=initj;
i=initi;
j=initj;

while t<=stacksize(2)



flags = zeros(1,8);

   if mamm(i+1,j-1)>=90
      if mamm(i+1,j-1)<=250
         flags(1)=1;
      end
   end
```

```
if mamm(i+1,j)>=90
       if mamm(i+1,j)<=250
      flags(2)=1;
  end
end

if mamm(i+1,j+1)>=90
   if mamm(i+1,j+1)<=250
      flags(3)=1;
   end
end

   if mamm(i,j+1)>=90
      if mamm(i,j+1)<=250
      flags(4)=1;
   end
end

if mamm(i-1,j+1)>=90
      if mamm(i-1,j+1)<=250
      flags(5)=1;
   end

if mamm(i-1,j)>=90
   if mamm(i-1,j)<=250
   flags(6)=1;
   end
end

if mamm(i-1,j-1)>=90
   if mamm(i-1,j-1)<=250
   flags(7)=1;
   end
end

   if mamm(i,j-1)>=90
    if mamm(i,j-1)<=250
      flags(8)=1;
   end
end



   if flags(1)==1
      coords=[i+1,j-1];
      flagcheck=ismember(coords,coordstack,'rows');

      if flagcheck==0

  stacksize=size(stack);
  stack(stacksize(2)+1)= mamm(i+1,j-1);
  coordstacksize=size(coordstack);
  coordstack((coordstacksize(1))+1,1)=(i+1);
  coordstack((coordstacksize(1))+1,2)=(j-1);
%  disp('n1');
        end %if
     end
```

```
if flags(2)==1
   coords=[i+1,j];
   flagcheck=ismember(coords,coordstack,'rows');

   if flagcheck==0

   stacksize=size(stack);
   stack(stacksize(2)+1)= mamm(i+1,j);
   coordstacksize=size(coordstack);
   coordstack((coordstacksize(1))+1,1)=(i+1);
   coordstack((coordstacksize(1))+1,2)=(j);
 %  disp('n2');
     end %if
   end



if flags(3)==1
   coords=[i+1,j+1];
   flagcheck=ismember(coords,coordstack,'rows');

   if flagcheck==0

   stacksize=size(stack);
   stack(stacksize(2)+1)= mamm(i+1,j+1);
   coordstacksize=size(coordstack);
   coordstack((coordstacksize(1))+1,1)=(i+1);
   coordstack((coordstacksize(1))+1,2)=(j+1);
%  disp('n3');
     end %if
   end


if flags(4)==1
   coords=[i,j+1];
   flagcheck=ismember(coords,coordstack,'rows');

   if flagcheck==0

   stacksize=size(stack);
   stack(stacksize(2)+1)= mamm(i,j+1);
   coordstacksize=size(coordstack);
   coordstack((coordstacksize(1))+1,1)=(i);
   coordstack((coordstacksize(1))+1,2)=(j+1);
   %disp('n4');
end %if
   end


if flags(5)==1
   coords=[i-1,j+1];
   flagcheck=ismember(coords,coordstack,'rows');

   if flagcheck==0

   stacksize=size(stack);
   stack(stacksize(2)+1)= mamm(i-1,j+1);
   coordstacksize=size(coordstack);
   coordstack((coordstacksize(1))+1,1)=(i-1);
```

```
    coordstack((coordstacksize(1))+1,2)=(j+1);
%   disp('n5');

end %if
    end



if flags(6)==1
    coords=[i-1,j];
    flagcheck=ismember(coords,coordstack,'rows');

    if flagcheck==0

    stacksize=size(stack);
    stack(stacksize(2)+1)= mamm(i-1,j);
    coordstacksize=size(coordstack);
    coordstack((coordstacksize(1))+1,1)=(i-1);
    coordstack((coordstacksize(1))+1,2)=(j);
%    disp('n6');

end %if
    end



if flags(7)==1
    coords=[i-1,j-1];
    flagcheck=ismember(coords,coordstack,'rows');

    if flagcheck==0

    stacksize=size(stack);
    stack(stacksize(2)+1)= mamm(i-1,j-1);
    coordstacksize=size(coordstack);
    coordstack((coordstacksize(1))+1,1)=(i-1);
    coordstack((coordstacksize(1))+1,2)=(j-1);
%   disp('n7');

end %if
    end



if flags(8)==1
    coords=[i,j-1];
    flagcheck=ismember(coords,coordstack,'rows');

    if flagcheck==0

    stacksize=size(stack);
      stack(stacksize(2)+1)= mamm(i,j-1);
      coordstacksize=size(coordstack);
      coordstack((coordstacksize(1))+1,1)=(i);
      coordstack((coordstacksize(1))+1,2)=(j-1);
 %   disp('n8');

   end %if
   end

disp(stacksize);
```

```
    t=t+1;
    i=coordstack(t,1);
    j=coordstack(t,2);

    %plot(coordstack(:,2),coordstack(:,1),'k.');

end %while

area=(stacksize(2)+1);
area
showfill;
```

Flitering and Smoothing

```
% Create desired frequency response
order=20;
cutoff=0.0001;
[f1,f2] = freqspace(order,'meshgrid');
d = find(f1.^2+f2.^2 < cutoff^2);
Hd = zeros(order);
Hd(d) = 1;
Hd = 1-Hd;  %hipass

h = fsamp2(Hd);

filtimage=filter2(h,av);

figure;
colormap(flag);
imagesc(filtimage);
```

```
% smoothing by 2D median averaging, neighbourhood 10 by 10  for use
after enhance2!!!!!!!!!!!!

fav=medfilt2(filtimage,[10 10]);



fav2=zeros(1384,1460);

for cc1=1:1384
   for cc2=1:1460
      if fav(cc1,cc2)>=1
```

```
            fav2(cc1,cc2)=0;
        else
            fav2(cc1,cc2)=1;
        end
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Conversion from Scan Dat File into Matix

```
% covert dat string into matrix.


fid = fopen ('Idoylel.dat','r');
a=fread(fid);
status=fclose(fid);
mamm=zeros(1752,1460);

    for c=1:1752
        b=a(((c-1)*1460)+1):(c*1460));
        e=b';
        mamm(c,:)=e;
    end
```

Curve Fitting

```


%%%%%%%%% CURVE FITTING LINEAR REGRESSION ALGORTIHM (version 2)
% and show plot with image

szcol=size(curvecols);
maxnumcol=szcol(2);

%%%%find upper point on curve
uplevel=min(curvecols(1:699));
Iup=find(curvecols(1:699)==uplevel);
Iupmax=max(Iup);

uptol=uplevel+100;

cca=699;

while curvecols(cca)>=uptol
```

```
    upperpoint=cca;
    cca=cca-1;
end



%uppercurve=find(curvecols(Iupmax:699)>uptol);
%upperpoint=min(uppercurve);


%%%%%%find lower point on curve
lowlevel=min(curvecols(700:maxnumcol));
Ilow=find(curvecols(700:maxnumcol)==lowlevel);

Ilow=(Ilow+699);
Ilowmin=min(Ilow);
lowtol=lowlevel+100;

ccc=700;

while curvecols(ccc)>=lowtol
    lowerpoint=ccc;
    ccc=ccc+1;
end


%lowercurve=find(curvecols(700:Ilowmin)>lowtol);
%lowerpoint=max(lowercurve);
%lowerpoint=lowerpoint+699;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


curve=polyfit((upperpoint:lowerpoint),curvecols(upperpoint:lowerpoint
),4);

yy=polyval(curve,1:maxnumcol);

axis ij;
colormap(gray);
hold on;
showseg;
plot(yy,1:maxnumcol,'r')
```

Segmentation

```
% segmetantion heuristic method version 2

%identify highest row of lower black section

gradblack=gradient(mamm(:,1460));
mingradblack=min(gradient(mamm(:,1460)));
```

```
Iblack=find(gradblack==mingradblack);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%find minimum of gradients at rows = 300 and 900

grada=gradient(mamm(300,:));
gradb=gradient(mamm(900,:));


mingrada=min(gradient(mamm(300,:)));
mingradb=min(gradient(mamm(900,:)));

IA=min(find(grada==mingrada));  %%%%%%
IB=min(find(gradb==mingradb)); %to make dy and dx same dimension -
added 27/11/00


dy=-600;

dx=IA-IB;

if dx==0;

   for r=1:Iblack
      edgecols(1,r)=IA;
   end
else

   m=(dy/dx);

   c=(300-(m*IA));

   edgecols=zeros(1,Iblack);

   for r=1:Iblack
      edgecols(1,r)=((r-c)/m);
   end
end

rdgecols=round(edgecols);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%find grey level of radiation. using row= 700

radiation=min(mamm(700,:)); %%%%


temp=min(mamm(:,:)');
temp_max=max(temp(1:Iblack));

tolerance=temp_max+1;    %%%%%

size(temp);
rad=ones(1,ans(2))*(radiation+15);   %%%%
```

```
rad_new=max(temp',rad');

rad_new=rad_new+1; %%%%% test

for counter=1:Iblack

   mammrowtemp=mamm(counter,(rdgecols(counter)):1460);

   Iradbits=find(mammrowtemp<rad_new(counter));%%%%
   Iradbits=(Iradbits+((rdgecols(1,counter)-1)));
   check=isempty(Iradbits);
   if check==1;
      curvecols(counter)=rdgecols(counter);
   else
      curvecols(counter)=min(Iradbits);

   end

end
```

Compare Shape

```
clear
load jtinsle1
load jtinsle2

nslice1=zeros(2,101);
nslice2=zeros(2,101);

shift1=slice1(:,1);
shift2=slice2(:,1);

for i=1:101
   nslice1(:,i)=slice1(:,i)-shift1;
   nslice2(:,i)=slice2(:,i)-shift2;
end

max1=max(slice1(2,:));
max2=max(slice2(2,:));

for i=1:101
   nslice1(2,i)=slice1(2,i)/max1;
   nslice2(2,i)=slice2(2,i)/max2;
end

for iii=1:101
nsliceflip(iii)=nslice2(2,102-iii);
end

hold on
plot(nslice1(2,:),nslice1(1,:));
plot(nslice2(2,:),nslice2(1,:),'r');
plot(nsliceflip,nslice2(1,:),'g');

ysl=(nslice1(2,:))-(nsliceflip);
for ii=1:101
   ysl(ii)=sqrt((ysl(ii))^2);
end


sumysl=sum(ysl);
```

```
save jtinslescaledistflip sumysl
disp(sumysl)


***********
%%%%%%%%%% find 100 shape coordinates on curve %%%%%%%

%%%load image******
clear
fid = fopen ('Vtrubyl.dat','r');
a=fread(fid);
status=fclose(fid);
mamm=zeros(1752,1460);

   for c=1:1752
      b=a(((c-1)*1460)+1):(c*1460));
      e=b';
      mamm(c,:)=e;
   end
   %%%%%%%%%%%%%%%%%%%%

   segfit2;


  limit=size(yyyy);
hlimit=limit/2;

lim=limit(2);

halflimitemp=hlimit(2);
halflim=round(halflimitemp);

for count=1:halflim
ydiffs=yyyy(count)-yy(count);
sqydiffs(count)=sqrt(ydiffs^2);
end

topmin=min(sqydiffs);
JA=find(sqydiffs==min(sqydiffs));

for count=halflim:lim
ydiffs=yyyy(count)-yy(count);
sqydiffs(count)=sqrt(ydiffs^2);
end

sqydiffslatter=sqydiffs(halflim:lim);
IBtemp=find(sqydiffslatter==min(sqydiffslatter));
botmin=min(sqydiffslatter);
JB=IBtemp+(halflim-1);

 %%%%%%%crossover check %%%%%%%

if (topmin<2)&(botmin<2);
```

```
tipdiff=JB-JA; %difference in vertical x over points

   slicesize=tipdiff/100;

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   for slicenum=1:101
      xslice(slicenum)=JA+(slicesize*(slicenum-1));
   end

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



   yslice=polyval(curve,xslice);

   save Xslice xslice

   save Yslice yslice



else

   disp('bad fit! initiating curve fit with order 4 points');

   if topmin<2;
      showcurve2pluslo;
      disp('adding ord 4 lower');
      showline;

   else
      if botmin<2;
         showcurve2plushi;
         disp('adding ord 4 upper');
         showline;
      else
         showcurve2plus;
         disp('adding ord 4 both');
         showline;
      end
   end




   %%%%%%% Find crossover points of curve and line for limits of
Intergration%%%%%%
limit=size(yyyy);
hlimit=limit/2;

lim=limit(2);

halflimitemp=hlimit(2);
halflim=round(halflimitemp);

for count=1:halflim
ydiffs=yyyy(count)-yy(count);
sqydiffs(count)=sqrt(ydiffs^2);
end
```

```
%topmin=min(sqydiffs);
sqydiffslower=sqydiffs(1:halflim);
JA=find(sqydiffslower==min(sqydiffslower));

for count=halflim:lim
ydiffs=yyyy(count)-yy(count);
sqydiffs(count)=sqrt(ydiffs^2);
end
sqydiffslatter=sqydiffs(halflim:lim);
IBtemp=find(sqydiffslatter==min(sqydiffslatter));
%botmin=min(sqydiffslatter);
JB=IBtemp+(halflim-1);

   %%%%%%%%%%%%%%%%%%

   tipdiff=JB-JA; %difference in vertical x over points

   slicesize=tipdiff/100;

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for slicenum=1:101
   xslice(slicenum)=JA+(slicesize*(slicenum-1));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

yslice=polyval(curve,xslice);

save Xslice xslice

save Yslice yslice

end

   clear



%%%load next image*******

fid = fopen ('Vtrubyr.dat','r');
a=fread(fid);
status=fclose(fid);
mamm=zeros(1752,1460);

   for c=1:1752
      b=a(((c-1)*1460)+1):(c*1460));
      e=b';
      mamm(c,:)=e;
   end
   %%%%%%%%%%%%%%%%%%%%

   segfit2;

limit=size(yyyy);
```

```
hlimit=limit/2;

lim=limit(2);

halflimitemp=hlimit(2);
halflim=round(halflimitemp);

for count=1:halflim
ydiffs=yyyy(count)-yy(count);
sqydiffs(count)=sqrt(ydiffs^2);
end

topmin=min(sqydiffs);
JA=find(sqydiffs==min(sqydiffs));

for count=halflim:lim
ydiffs=yyyy(count)-yy(count);
sqydiffs(count)=sqrt(ydiffs^2);
end

sqydiffslatter=sqydiffs(halflim:lim);
IBtemp=find(sqydiffslatter==min(sqydiffslatter));
botmin=min(sqydiffslatter);
JB=IBtemp+(halflim-1);

   %%%%%%%%%%%%%%%%%%%
 %%%%%%%crossover check %%%%%%%

if (topmin<2)&(botmin<2);


   tipdiff=JB-JA; %difference in vertical x over points

   slicesize=tipdiff/100;

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   for slicenum=1:101
      xslice2(slicenum)=JA+(slicesize*(slicenum-1));
   end

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



   yslice2=polyval(curve,xslice2);

   save Xslice2 xslice2

   save Yslice2 yslice2

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5


else

   disp('bad fit! initiating curve fit with order 4 points');

   if topmin<2;
      showcurve2pluslo;
      disp('adding ord 4 lower');
```

```
      showline;

   else
      if botmin<2;
         showcurve2plushi;
         disp('adding ord 4 upper');
         showline;
      else
         showcurve2plus;
         disp('adding ord 4 both');
         showline;
      end
   end



   %%%%%%% Find crossover points of curve and line for limits of
Intergration%%%%%%
limit=size(yyyy);
hlimit=limit/2;

lim=limit(2);

halflimitemp=hlimit(2);
halflim=round(halflimitemp);

for count=1:halflim
ydiffs=yyyy(count)-yy(count);
sqydiffs(count)=sqrt(ydiffs^2);
end

%topmin=min(sqydiffs);
sqydiffslower=sqydiffs(1:halflim);
JA=find(sqydiffslower==min(sqydiffslower));

for count=halflim:lim
ydiffs=yyyy(count)-yy(count);
sqydiffs(count)=sqrt(ydiffs^2);
end

sqydiffslatter=sqydiffs(halflim:lim);
IBtemp=find(sqydiffslatter==min(sqydiffslatter));
%botmin=min(sqydiffslatter);
JB=IBtemp+(halflim-1);

   %%%%%%%%%%%%%%%%%%


   tipdiff=JB-JA; %difference in vertical x over points

   slicesize=tipdiff/100;

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   for slicenum=1:101
      xslice2(slicenum)=JA+(slicesize*(slicenum-1));
   end

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    yslice2=polyval(curve,xslice2);
    save Xslice2 xslice2
    save Yslice2 yslice2
end
```

```
load xslice
load yslice
slice1=[xslice; yslice];
slice2=[xslice2; yslice2];
%ysltemp=yslice-yslice2;
%xsltemp=xslice-xslice2;
%for c=1:101
   %ysl(c)=sqrt((ysltemp(c))^2);
   %xsl(c)=sqrt((xsltemp(c))^2);
%end

%sumysl=sum(ysl);
%sumxsl=sum(xsl);

save Vtruby1 slice1

save Vtruby2 slice2
```

Genetic Algorithm

```
% test function CSF neural network SSE of 75 samples from Iris
database
% best are smaller SSE values
% all chomosome numbers between -5 and 5

%target =0;
irisdata;
p=(inpatterns(1:75,:))';
pop=((rand(10,81)-0.5)*10);
gen=1;

while gen <=50000

%for i=1:10;
   %fitness(i)=sum(pop(i,:));
%end
%%%%%%%% nnet translation and fitness %%%%%%%%%

for z=1:10;
   w1long=pop(z,1:24);
```

```
clong=pop(z,25:48);
omlong=(((pop(z,49:54))+5)*0.1)*(pi));
b1long=pop(z,55:60);
w2long=pop(z,61:78);
b2long=pop(z,79:81);

w1=reshape(w1long,6,4);
c=reshape(clong,6,4);
b1=b1long';
 om=omlong;
 w2=reshape(w2long,3,6);
 b2=b2long';

 [hiddenout,outout]=simcon(c,p,w1,om,b1,w2,b2);

 sqdiffs=(targets(1:75,:)-outout').^2;

for zz=1:75
   sse(zz)=sum(sqdiffs(zz,:));
end

fitness(z)=1/(sum(sse));

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[kingfitness,king]=max(fitness);

others=fitness;
others(king)=[];

[queenfitness,queentemp]=max(others);

for j=1:10
   if fitness(j)==queenfitness
      queen=j;
   end
end


kingchromo=pop(king,:);
queenchromo=pop(queen,:);

ranum1=rand*81;
ranum2=rand*81;
   if ranum1 > ranum2
      xp2=ranum1;
      xp1=ranum2;
      else
         xp2=ranum2;
         xp1=ranum1;
   end
rxp1=round(xp1);
rxp2=round(xp2);


andychromo=[kingchromo(1:rxp1) queenchromo(rxp1+1:rxp2)
kingchromo(rxp2+1:81)];
```

```
eddychromo=[queenchromo(1:rxp1) kingchromo(rxp1+1:rxp2)
queenchromo(rxp2+1:81)];

mutantselect1=round(rand(1,81)*100);
mutantselect2=round(rand(1,81)*100);


for mutantcount1=1:81
   if mutantselect1(1,mutantcount1)==50
      mutanttemp1=andychromo(mutantcount1)+((rand(1,1)-0.5)*2);
      if mutanttemp1 <=5
         if mutanttemp1 >=-5
            andychromo(mutantcount1)=mutanttemp1;
         end
      end
   end
end


for mutantcount2=1:81
   if mutantselect2(1,mutantcount2)==50
      mutanttemp2=eddychromo(mutantcount2)+((rand(1,1)-0.5)*2);
      if mutanttemp2 <=5
         if mutanttemp2 >=-5
            eddychromo(mutantcount2)=mutanttemp2;
         end
      end
   end
end


%%%%%%%next gen fitness eval for csf**********

   andyw1long=andychromo(1:24);
   andyclong=andychromo(25:48);
   andyomlong=(((andychromo(49:54))+5)*0.1)*(pi));
   andyb1long=andychromo(55:60);
   andyw2long=andychromo(61:78);
   andyb2long=andychromo(79:81);

   andyw1=reshape(andyw1long,6,4);
   andyc=reshape(andyclong,6,4);
   andyb1=andyb1long';
   andyom=andyomlong;
   andyw2=reshape(andyw2long,3,6);
   andyb2=andyb2long';


[hiddenout,andyoutout]=simcon(andyc,p,andyw1,andyom,andyb1,andyw2,and
yb2);

   sqdiffs=(targets(1:75,:)-andyoutout').^2;

  for zzz=1:75
     sse(zzz)=sum(sqdiffs(zzz,:));
  end

  andyfitness=1/(sum(sse));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   eddyw1long=eddychromo(1:24);
   eddyclong=eddychromo(25:48);
```

```
   eddyomlong=((((eddychromo(49:54))+5)*0.1)*(pi));
   eddyb1long=eddychromo(55:60);
   eddyw2long=eddychromo(61:78);
   eddyb2long=eddychromo(79:81);

   eddyw1=reshape(eddyw1long,6,4);
   eddyc=reshape(eddyclong,6,4);
   eddyb1=eddyb1long';
   eddyom=eddyomlong;
   eddyw2=reshape(eddyw2long,3,6);
   eddyb2=eddyb2long';


[hiddenout,andyoutout]=simcon(eddyc,p,eddyw1,eddyom,eddyb1,eddyw2,edd
yb2);

   sqdiffs=(targets(1:75,:)-andyoutout').^2;

  for zzz=1:75
     sse(zzz)=sum(sqdiffs(zzz,:));
  end

  eddyfitness=1/(sum(sse));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%andyfitness=sum(andychromo);
%eddyfitness=sum(eddychromo);

if andyfitness>=eddyfitness

   newking1fit=andyfitness;
   newking1=andychromo;
   newking2fit=eddyfitness;
   newking2=eddychromo;
else
   newking1fit=eddyfitness;
   newking1=eddychromo;
   newking2fit=andyfitness;
   newking2=eddychromo;
end

if newking1fit>=queenfitness
      if newking2fit>=kingfitness
         pop(king,:)=newking1;
         pop(queen,:)=newking2;
      else
           if newking1fit>=kingfitness
              pop(queen,:)=pop(king,:);
              pop(king,:)=newking1;

           else
              pop(queen,:)=newking1;
           end
      end

end



newpoptemp=((rand(8,81)-0.5)*10);
newpop=[newpoptemp;pop(king,:);pop(queen,:)];
```

```
pop=[];
pop=newpop;
%target=kingfitness;
results(gen)=kingfitness;
gen=gen+1;

%hold on;
%xlabel ('generations');
%ylabel ('fitness value');
%plot(1:3908,results,'k.');

%disp(gen);
disp(kingfitness);

end
```

EBP

```
function
[w1,b1,w2,b2,w3,b3,i,tr]=tbpx3(w1,b1,f1,w2,b2,f2,w3,b3,f3,p,t,tp)
%TBPX3 Train 3-layer feed-forward network w/fast backpropagation.
%
%  [W1,B1,W2,B2,W3,B3,TE,TR] =
TBPX3(W1,B2,F1,W1,B1,F2,W3,B3,F3,P,T,TP)
%    Wi - Weight matrix for the ith layer.
%    Bi - Bias vector for the ith layer.
%    Fi - Transfer function (string) for the ith layer.
%    P  - RxQ matrix of input vectors.
%    T  - SxQ matrix of target vectors.
%    TP - Training parameters (optional).
%  Returns:
%    Wi - new weights.
%    Bi - new biases.
%    TE - the actual number of epochs trained.
%    TR - training record: [row of errors]
%
%  Training parameters are:
%    TP(1) - Epochs between updating display, default = 25.
%    TP(2) - Maximum number of epochs to train, default = 1000.
%    TP(3) - Sum-squared error goal, default = 0.02.
%    TP(4) - Learning rate, 0.01.
%    TP(5) - Learning rate increase, default = 1.05.
%    TP(6) - Learning rate decrease, default = 0.7.
%    TP(7) - Momentum constant, default = 0.9.
%    TP(8) - Maximum error ratio, default = 1.04.
%  Missing parameters and NaN's are replaced with defaults.

% Mark Beale, 1-31-92
% Revised 12-15-93, MB

if nargin < 11,error('Not enough arguments.');end

% TRAINING PARAMETERS
if nargin == 11, tp = []; end
tp = nndef(tp,[25 1000 0.02 0.01 1.05 0.7 0.9 1.04]);
df = tp(1);
me = tp(2);
eg = tp(3);
lr = tp(4);
```

```
im = tp(5);
dm = tp(6);
mc = tp(7);
er = tp(8);
df1 = feval(f1,'delta');
df2 = feval(f2,'delta');
df3 = feval(f3,'delta');

dw1 = w1*0;
db1 = b1*0;
dw2 = w2*0;
db2 = b2*0;
dw3 = w3*0;
db3 = b3*0;
MC = 0;

% PRESENTATION PHASE
a1 = feval(f1,w1*p,b1);
a2 = feval(f2,w2*a1,b2);
a3 = feval(f3,w3*a2,b3);
e = t-a3;
SSE = sumsqr(e);

% TRAINING RECORD
tr = zeros(2,me+1);
tr(1:2,1) = [SSE; lr];

% PLOTTING FLAG
[r,q] = size(p);
[s,q] = size(t);
plottype = (max(r,s) == 1) & 0;

% PLOTTING
newplot;
message = sprintf('TRAINBPX: %%g/%g epochs, lr = %%g, SSE =
%%g.\n',me);
fprintf(message,0,lr,SSE)
if plottype
  h = plotfa(p,t,p,a3);
else
  h = plottr(tr(1:2,1),eg);
end

% BACKPROPAGATION PHASE
d3 = feval(df3,a3,e);
d2 = feval(df2,a2,d3,w3);
d1 = feval(df1,a1,d2,w2);

for i=1:me

  % CHECK PHASE
  if SSE < eg, i=i-1; break, end

  % LEARNING PHASE
  [dw1,db1] = learnbpm(p,d1,lr,MC,dw1,db1);
  [dw2,db2] = learnbpm(a1,d2,lr,MC,dw2,db2);
  [dw3,db3] = learnbpm(a2,d3,lr,MC,dw3,db3);
  MC = mc;
  new_w1 = w1 + dw1; new_b1 = b1 + db1;
  new_w2 = w2 + dw2; new_b2 = b2 + db2;
  new_w3 = w3 + dw3; new_b3 = b3 + db3;
```

```
% PRESENTATION PHASE
new_a1 = feval(f1,new_w1*p,new_b1);
new_a2 = feval(f2,new_w2*new_a1,new_b2);
new_a3 = feval(f3,new_w3*new_a2,new_b3);
new_e = t-new_a3;
new_SSE = sumsqr(new_e);

% MOMENTUM & ADAPTIVE LEARNING RATE PHASE
if new_SSE > SSE*er
  lr = lr * dm;
  MC = 0;
else
  if new_SSE < SSE
    lr = lr * im;
    end
  w1 = new_w1; b1 = new_b1; a1 = new_a1;
  w2 = new_w2; b2 = new_b2; a2 = new_a2;
  w3 = new_w3; b3 = new_b3; a3 = new_a3;
  e = new_e; SSE = new_SSE;

  % BACKPROPAGATION PHASE
  d3 = feval(df3,a3,e);
  d2 = feval(df2,a2,d3,w3);
  d1 = feval(df1,a1,d2,w2);
end

% TRAINING RECORD
tr(1:2,i+1) = [SSE; lr];

% PLOTTING
if rem(i,df) == 0
  fprintf(message,i,lr,SSE)
  if plottype
    delete(h);
    h = plot(p,a3);
  else
    h = plottr(tr(1:2,1:(i+1)),eg,h);
    end
  end
end

% TRAINING RECORD
tr = tr(1:2,1:(i+1));

% PLOTTING
if rem(i,df) ~= 0
  fprintf(message,i,lr,SSE)
  if plottype
    delete(h);
    plot(p,a3);
  else
    plottr(tr,eg,h);
  end
end

% WARNINGS
if SSE > eg
  disp(' ')
  disp('TRAINBPX: Network error did not reach the error goal.')
  disp('  Further training may be necessary, or try different')
```

```
  disp('  initial weights and biases and/or more hidden neurons.')
  disp(' ')
end
```

OLS

```
function [w1,b1,w2,b2,k,tr] = solverb(p,t,tp)
%SOLVERB Design radial basis network.
%
%  [W1,B1,W2,B2,TE,TR] = SOLVERB(P,T,DP)
%    P - RxQ matrix of Q input vectors.
%    T - SxQ matrix of Q target vectors.
%    DP - Design parameters (optional).
%  Returns:
%    W1 - S1xR weight matrix for radial basis layer.
%    B1 - S1x1 bias vector for radial basis layer.
%    W2 - S2xS1 weight matrix for linear layer.
%    B2 - S2x1 bias vector for linear layer.
%    NR - the number of radial basis neurons used.
%    TR - training record: [row of errors]
%
%  Design parameters are:
%    DP(1) - Iterations between updating display, default = 25.
%    DP(2) - Maximum number of neurons, default = # vectors in P.
%    DP(3) - Sum-squared error goal, default = 0.02.
%    DP(4) - Spread of radial basis functions, default = 1.0.
%  Missing parameters and NaN's are replaced with defaults.
%
%  See also NNSOLVE, RADBASIS, SIMRB, SOLVERB.


if nargin < 2, error('Not enough input arguments'),end

% TRAINING PARAMETERS
if nargin == 2, tp = []; end
[r,q] = size(p);
tp = nndef(tp,[25 q 0.02 1]);
df = tp(1);
eg = tp(3);
b = sqrt(-log(.5))/tp(4);
[s2,q] = size(t);
mn = min(q,tp(2));

% PLOTTING FLAG
plottype = max(r,s2) == 1;

% RADIAL BASIS LAYER OUTPUTS
P = radbas(dist(p',p)*b);
PP = sum(P.*P)';
d = t';
dd = sum(d.*d)';

% CALCULATE "ERRORS" ASSOCIATED WITH VECTORS
e = ((P' * d)' .^ 2) ./ (dd * PP');

% PICK VECTOR WITH MOST "ERROR"
pick = nnfmc(e);
```

```
used = [];
left = 1:q;
W = P(:,pick);
P(:,pick) = []; PP(pick,:) = [];
e(:,pick) = [];
used = [used left(pick)];
left(pick) = [];

% CALCULATE ACTUAL ERROR
w1 = p(:,used)';
a1 = radbas(dist(w1,p)*b);
[w2,b2] = solvelin(a1,t);
a2 = purelin(w2*a1,b2);
sse = sumsqr(t-a2);

% TRAINING RECORD
tr = zeros(1,mn);
tr(1) = sse;

% PLOTTING
newplot;
if plottype
  h = plotfa(p,t,p,a2);
else
  h = ploterr(tr(1),eg);
end

for k = 1:(mn-1)

  % CHECK ERROR
  if (sse < eg), break, end

  % CALCULATE "ERRORS" ASSOCIATED WITH VECTORS

  wj = W(:,k);

  %---- VECTOR CALCULATION
  a = wj' * P / (wj'*wj);
  P = P - wj * a;
  PP = sum(P.*P)';
%if any(any(PP == 0))
%  disp('PP has a 0')
%  keyboard
%end
  e = ((P' * d)' .^ 2) ./ (dd * PP');
.
  % PICK VECTOR WITH MOST "ERROR"
  pick = nnfmc(e);
  W = [W, P(:,pick)];
  P(:,pick) = []; PP(pick,:) = [];
  e(:,pick) = [];
  used = [used left(pick)];
  left(pick) = [];

  % CALCULATE ACTUAL ERROR
  w1 = p(:,used)';
  a1 = radbas(dist(w1,p)*b);
  [w2,b2] = solvelin(a1,t);
  a2 = purelin(w2*a1,b2);
  sse = sumsqr(t-a2);
```

```
% TRAINING RECORD
tr(k+1) = sse;

% PLOTTING
if rem(k,df) == 0
  if plottype
    delete(h);
    h = plot(p,a2,'m');
    drawnow;
  else
    h = ploterr(tr(1:(k+1)),eg,h);
  end
end

[S1,R] = size(w1);
b1 = ones(S1,1)*b;

% TRAINING RECORD
tr = tr(1:(k+1));

% PLOTTING
if rem(k,df) ~= 0
  if plottype
    delete(h);
    plot(p,a2,'m');
    drawnow;
  else
    ploterr(tr,eg,h);
  end
end

% WARNINGS
if sse > eg
  disp(' ')
  disp('SOLVERB: Network error did not reach the error goal.')
  disp('  More neurons may be necessary, or try using a')
  disp('  wider or narrower spread constant.')
  disp(' ')
end
```

Log likelihood error and weight decay

```
function [w1,b1,w2,b2,i,tr]=tbp2llwd(w1,b1,f1,w2,b2,f2,p,t,tp)
%TBP2 Train 2-layer feed-forward network w/backpropagation.
%
%  This function is obselete.
%  Use NNT2FF and TRAIN to update and train your network.

nntobsf('tbp2','Use NNT2FF and TRAIN to update and train your
network.')
mu=0.001;
%  [W1,B1,W2,B2,TE,TR] = TBP2(W1,B1,F1,W2,B2,F2,P,T,TP)
%    Wi - SixR weight matrix of ith layer.
%    Bi - Six1 bias vector of ith layer.
%    F  - Transfer function (string) of ith layer.
%    P  - RxQ matrix of input vectors.
```

```
%    T  - S2xQ matrix of target vectors.
%    TP - Training parameters (optional).
%  Returns:
%    Wi - new weights.
%    Bi - new biases.
%    TE - the actual number of epochs trained.
%    TR - training record: [row of errors]
%
%  Training parameters are:
%    TP(1) - Epochs between updating display, default = 25.
%    TP(2) - Maximum number of epochs to train, default = 1000.
%    TP(3) - Sum-squared error goal, default = 0.02.
%    TP(4) - Learning rate, 0.01.
%  Missing parameters and NaN's are replaced with defaults.

%  Mark Beale, 1-31-92
%  Revised 12-15-93, M.B.
% Copyright (c) 1992-1998 by The MathWorks, Inc.
% $Revision: 1.10 $

if nargin < 8,error('Not enough arguments.'),end

% TRAINING PARAMETERS
if nargin == 8, tp = []; end
tp = nndef(tp,[25 1000 0.02 0.01]);
df = tp(1);
me = tp(2);
eg = tp(3);
lr = tp(4);
df1 = feval(f1,'delta');
df2 = feval(f2,'delta');
sumw1=sum(sum(w1));
  sumw2=sum(sum(w2));

  sumw=(sumw1+sumw2);

  wdec=(sumsqr(sumw))/2;

% PRESENTATION PHASE
a1 = feval(f1,w1*p,b1);
a2 = feval(f2,w2*a1,b2);
e = t-a2;
e2 =e+(mu*wdec);
%e3 = e+1;
LLE= sum((t.*log(a2))+((1-t).*log(1-a2)));
LLE=sum(LLE);
LLE=-LLE;
LLE=LLE+(mu*wdec);

%LLE=t(1,:).*(log(a2(1,:)./t(1,:)));
  %LLEp2=t(2,:).*(log(a2(2,:)./t(2,:)));
  %LLE=LLEp1+LLEp2;
  %LLE=sum(LLE);
  %LLE=-LLE;
%LLE=LLE+(mu*wdec);

% PLOTTING FLAG
[r,q] = size(p);
[s2,q] = size(t);
plottype = max(r,s2) == 1;
```

```
% TRAINING RECORD
tr = zeros(1,me);
%tr(1) = LLE;

% PLOTTING
newplot;
message = sprintf('TRAINBP: %%g/%g epochs, LLE = %%g.\n',me);
fprintf(message,0,LLE)

if plottype
  h = plotfa(p,t,p,a2);
else
  h = ploterr2(tr(1),eg);
end

for cc=1:me

  % CHECK PHASE
  if LLE  < eg, cc=cc-1; break, end

  % BACKPROPAGATION PHASE

  d2 = feval(df2,a2,e2);
  d1 = feval(df1,a1,d2,w2);

  % LEARNING PHASE
  [dw1,db1] = learnbp(p,d1,lr);
  [dw2,db2] = learnbp(a1,d2,lr);

  w1 = w1 + dw1; b1 = b1 + db1;
  w2 = w2 + dw2; b2 = b2 + db2;

%%%%%VALIDATION%%%%%%

%[at1 at2]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');

% PRESENTATION PHASE
  a1 = feval(f1,w1*p,b1);
  a2 = feval(f2,w2*a1,b2);
  e = t-a2;
  e2 =e+(mu*wdec);
  %e3 = e+1;
  %SSE = sumsqr(e);

  sumw1=sum(sum(w1));
  sumw2=sum(sum(w2));

  sumw=(sumw1+sumw2);

  wdec2=(sumsqr(sumw))/2;

  %LLE=t(1,:).*(log(a2(1,:)./t(1,:)));
  %LLEp2=t(2,:).*(log(a2(2,:)./t(2,:)));
  %LLE=LLEp1+LLEp2;
  %LLE=sum(LLE);
  %LLE=-LLE;
%LLE=LLE+(mu*wdec2);
  LLE= sum((t.*log(a2))+((1-t).*log(1-a2)));
  LLE=sum(LLE);
```

```
LLE=-LLE;
LLE=LLE+(mu*wdec2);



% TRAINING RECORD
tr(cc+1) = LLE;

% PLOTTING
  if rem(cc,df) == 0
  fprintf(message,cc,LLE)
  disp(wdec2);
  if plottype
      delete(h);
      h = plot(LLE,a2);
      drawnow;
    else
      h = ploterr2(tr(1:(cc+1)),eg,h);
    end
  end
end

% TRAINING RECORD
tr = tr(1:(cc+1));

% PLOTTING
if rem(cc,df) ~= 0
  fprintf(message,cc,LLE)
disp(wdec);
  if plottype
    delete(h);
    plot(LLE,a2);
    drawnow;
  else
    ploterr2(tr,eg,h);
  end
end

% WARNINGS
if LLE > eg
  disp(' ')
  disp('TRAINBP: Network error did not reach the error goal.')
  disp('  Further training may be necessary, or try different')
  disp('  initial weights and biases and/or more hidden neurons.')
  disp(' ')
end
```

Alteration of Ordinal Inputs

```
%%%%%%%%%%%%%%%%%%%%%%%%%%% CONVERSION OF ORDINAL INPUTS INTO PDF
CURVE %%%%%%%%%%%%%

i=1;
j=1;
k=1;


for j=1:517
```

```
for i=1:6
   if P(i,j)==1
      peak=i;
      else
   end
end


   xtemp=1:6;
   gauss=normpdf(xtemp,peak,0.5);

   P(1:6,j)=gauss';

   for k=7:10
      if P(k,j)==1
         peak2=(k-6);
         else
      end
   end


      xtemp=1:4;
      gauss2=normpdf(xtemp,peak2,0.5);

      P(7:10,j)=gauss2';

end


for j=1:47

   for i=1:6
      if Ptest(i,j)==1
         peak=i;
         else
      end
   end


      xtemp=1:6;
      gauss=normpdf(xtemp,peak,0.5);

      Ptest(1:6,j)=gauss';


   for k=7:10
      if P(k,j)==1
         peak2=(k-6);
         else
      end
   end


      xtemp=1:4;
      gauss2=normpdf(xtemp,peak2,0.5);

      Ptest(7:10,j)=gauss2';

end
```

12 Fold cross validation output and ROC curve calculation

```
nntwarn off;
scuttdata;

outputs=zeros(564,2);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

targets=zeros(564,1);



for iiii=1:564
   if caorno(iiii)==1
      targets(iiii)=0.9;
      %targets(iiii,2)=0.1;
   else
      targets(iiii)=0.1;
      %targets(iiii,2)=0.9;
   end
end

outputs(:,2)=targets;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%


relvol=zeros(564,1);

for ii=1:564
   difftemp=(rvol(ii)-lvol(ii));
   sumtemp=(rvol(ii)+lvol(ii));
   relvol(ii)=(sqrt((difftemp)^2))/sumtemp;
end



fh=zeros(564,4);

for iii=1:564
   if famhis(iii)==0
      fh(iii,1)=1;
   else
      if famhis(iii)==1
         fh(iii,2)=1;
      else
         if famhis(iii)==2
            fh(iii,3)=1;
         else
            fh(iii,4)=1;
         end
      end
   end
end


lpch=zeros(564,4);
```

```
for iii=1:564
   if lparench(iii)==1
      lpch(iii,1)=1;
   else
      if lparench(iii)==2
         lpch(iii,2)=1;
      else
         if lparench(iii)==3
            lpch(iii,3)=1;
         else
            lpch(iii,4)=1;
         end
      end
   end
end


rpch=zeros(564,4);

for iii=1:564
   if rparench(iii)==1
      rpch(iii,1)=1;
   else
      if rparench(iii)==2
         rpch(iii,2)=1;
      else
         if rparench(iii)==3
            rpch(iii,3)=1;
         else
            rpch(iii,4)=1;
         end
      end
   end
end


agm=zeros(564,6);

for iii=1:564
   if ageatmen(iii)<=11
      agm(iii,1)=1;
   else
      if ageatmen(iii)==12
         agm(iii,2)=1;
      else
         if ageatmen(iii)==13
            agm(iii,3)=1;
         else
            if ageatmen(iii)==14
               agm(iii,4)=1;
            else
               if ageatmen(iii)==15
               agm(iii,5)=1;
               else
                  agm(iii,6)=1;
               end
            end
         end
      end
   end
end
```

```
   end
end



pl=zeros(564,1);

for iii=1:564
   if pill(iii)==1
      pl(iii)=0;
   else
      pl(iii)=1;
   end
end


%%%%%%%%%%%%%%%%%%% 1 to 47 section
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%





Ptest=zeros(20,47);


agmxtest=agm(1:47,:)';
fhxtest=fh(1:47,:)';
lpchxtest=lpch(1:47,:)';
rpchxtest=rpch(1:47,:)';
relvolxtest=relvol(1:47)';
plxtest=pl(1:47,:)';

Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;



load mlp0;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');


outputs1=outputs1';
outputs(1:47,1)=outputs1;

clear W1 B1 W2 B2;
clear Ptest;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ptest=zeros(20,47);

agmxtest=agm(48:94,:)';
fhxtest=fh(48:94,:)';
```

```
lpchxtest=lpch(48:94,:)';
rpchxtest=rpch(48:94,:)';
relvolxtest=relvol(48:94)';
plxtest=pl(48:94,:)';

Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;



load 47mlp;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');


outputs1=outputs1';
outputs(48:94,1)=outputs1;

clear W1 B1 W2 B2;
clear Ptest;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



Ptest=zeros(20,47);

agmxtest=agm(95:141,:)';
fhxtest=fh(95:141,:)';
lpchxtest=lpch(95:141,:)';
rpchxtest=rpch(95:141,:)';
relvolxtest=relvol(95:141)';
plxtest=pl(95:141,:)';

Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;

load 94mlp;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');


outputs1=outputs1';
outputs(95:141,1)=outputs1;

clear W1 B1 W2 B2;
clear Ptest;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



Ptest=zeros(20,47);

agmxtest=agm(142:188,:)';
fhxtest=fh(142:188,:)';
lpchxtest=lpch(142:188,:)';
rpchxtest=rpch(142:188,:)';
relvolxtest=relvol(142:188)';
plxtest=pl(142:188,:)';



Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;

load 141mlp;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');


outputs1=outputs1';
outputs(142:188,1)=outputs1;

clear W1 B1 W2 B2;
clear Ptest;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



Ptest=zeros(20,47);

agmxtest=agm(189:235,:)';
fhxtest=fh(189:235,:)';
lpchxtest=lpch(189:235,:)';
rpchxtest=rpch(189:235,:)';
relvolxtest=relvol(189:235)';
plxtest=pl(189:235,:)';



Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;

load 188mlp;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');
```

```
outputs1=outputs1';
outputs(189:235,1)=outputs1;

clear W1 B1 W2 B2;
clear Ptest;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



Ptest=zeros(20,47);

agmxtest=agm(236:282,:)';
fhxtest=fh(236:282,:)';
lpchxtest=lpch(236:282,:)';
rpchxtest=rpch(236:282,:)';
relvolxtest=relvol(236:282)';
plxtest=pl(236:282,:)';



Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;

load 235mlp;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');


outputs1=outputs1';
outputs(236:282,1)=outputs1;

clear W1 B1 W2 B2;
clear Ptest;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



Ptest=zeros(20,47);

agmxtest=agm(283:329,:)';
fhxtest=fh(283:329,:)';
lpchxtest=lpch(283:329,:)';
rpchxtest=rpch(283:329,:)';
relvolxtest=relvol(283:329)';
plxtest=pl(283:329,:)';



Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
```

```
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;

load 282mlp;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');


outputs1=outputs1';
outputs(283:329,1)=outputs1;

clear W1 B1 W2 B2;
clear Ptest;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Ptest=zeros(20,47);

agmxtest=agm(330:376,:)';
fhxtest=fh(330:376,:)';
lpchxtest=lpch(330:376,:)';
rpchxtest=rpch(330:376,:)';
relvolxtest=relvol(330:376)';
plxtest=pl(330:376,:)';



Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;

load 329mlp;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');


outputs1=outputs1';
outputs(330:376,1)=outputs1;

clear W1 B1 W2 B2;
clear Ptest;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Ptest=zeros(20,47);

agmxtest=agm(377:423,:)';
```

```
fhxtest=fh(377:423,:)';
lpchxtest=lpch(377:423,:)';
rpchxtest=rpch(377:423,:)';
relvolxtest=relvol(377:423)';
plxtest=pl(377:423,:)';



Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;

load 376mlp;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');


outputs1=outputs1';
outputs(377:423,1)=outputs1;

clear W1 B1 W2 B2;
clear Ptest;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Ptest=zeros(20,47);

agmxtest=agm(424:470,:)';
fhxtest=fh(424:470,:)';
lpchxtest=lpch(424:470,:)';
rpchxtest=rpch(424:470,:)';
relvolxtest=relvol(424:470)';
plxtest=pl(424:470,:)';



Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;

load 423mlp;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');


outputs1=outputs1';
outputs(424:470,1)=outputs1;

clear W1 B1 W2 B2;
```

```
clear Ptest;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Ptest=zeros(20,47);

agmxtest=agm(471:517,:)';
fhxtest=fh(471:517,:)';
lpchxtest=lpch(471:517,:)';
rpchxtest=rpch(471:517,:)';
relvolxtest=relvol(471:517)';
plxtest=pl(471:517,:)';

Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;

load 470mlp;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');

outputs1=outputs1';
outputs(471:517,1)=outputs1;

clear W1 B1 W2 B2;
clear Ptest;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Ptest=zeros(20,47);

agmxtest=agm(518:564,:)';
fhxtest=fh(518:564,:)';
lpchxtest=lpch(518:564,:)';
rpchxtest=rpch(518:564,:)';
relvolxtest=relvol(518:564)';
plxtest=pl(518:564,:)';

Ptest(1:6,:)=agmxtest;
Ptest(7:10,:)=fhxtest;
Ptest(11:14,:)=lpchxtest;
Ptest(15:18,:)=rpchxtest;
Ptest(19,:)=relvolxtest;
Ptest(20,:)=plxtest;
```

```
load 517mlp2;

[a,outputs1]=simuff(Ptest,W1,B1,'logsig',W2,B2,'logsig');


outputs1=outputs1';
outputs(518:564,1)=outputs1;

clear W1 B1 W2 B2;
clear Ptest;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


low=zeros(1,282);
high=zeros(1,282);

l=1;
h=1;



for i=1:564
   if outputs(i,2)==0.1
      low(l)=outputs(i,1);
      l=l+1;
   else
      high(h)=outputs(i,1);
      h=h+1;
   end
end

low=low';
high=high';


n=hist(low,100);
m=hist(high,100);
hold on
plot(n);
plot(m,'g-');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
cc1=1;
cc2=1;
cc3=1;
cc4=1;


accuracy=zeros(1,101);
sensitivity=zeros(1,101);
specitvity=zeros(1,101);
oneminusspecitivity=zeros(1,101);
```

```
for k=0:100
      thresh=(k/100);
      %disp(thresh);
      for j=1:282
         if high(j)>thresh
            TP(cc1)=high(j);
            cc1=cc1+1;
         else
            FN(cc2)=high(j);
            cc2=cc2+1;
         end
      end
      for z=1:282
         if low(z)>thresh
            FP(cc3)=low(z);
            cc3=cc3+1;
         else
            TN(cc4)=low(z);
            cc4=cc4+1;
         end
      end

      NTP=cc1-1;
      NFN=cc2-1;
      NFP=cc3-1;
      NTN=cc4-1;

      accuracy(k+1)=(NTP+NTN)/564;
      sensitivity(k+1)=NTP/(NTP+NFN);
      specitivity(k+1)=NTN/(NTN+NFP);

      oneminusspecitivity(k+1)=1-(specitivity(k+1));

      cc1=1;
cc2=1;
cc3=1;
cc4=1;



   end


   figure;
   %axis([0 1, 0 1]);
   plot(oneminusspecitivity,sensitivity);
```