IMPROVED CLASSIFICATION IN FLAT NETWORKS

by

MADHU GANNAPAL

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2010

ACKNOWLEDGEMENTS

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

The EE coursework at UTA and my undergraduate institute SJCE in India provided the fillip and toolkit to weave the pieces of this work together. Hence a sincere thanks to all the teachers who selflessly strive to spread education to whom I dedicate this work. Thanks to Dr. Qilian Liang and Dr. Soontorn Oraintara for a patient review of the thesis work.

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

April 15, 2010

ABSTRACT

IMPROVED CLASSIFICATION IN FLAT NETWORKS

Madhu Gannapal, M.S.

The University of Texas at Arlington, 2010

Supervising Professor: Michael T. Manry

It is shown that optimal flat networks can be found as solutions to least squares problems. An algorithm is presented to improve existing classifier training methods by changing the desired outputs. The algorithm is based on minimum probability of error. The algorithm's performance is compared with those of other algorithms including the Bayes Gaussian classifier. The Convergence of training and the effects of outliers are analyzed in all the algorithms presented here.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Neural Networks and Research

Multilayer perceptron (MLP) neural nets [1] have proved useful in many approximation and classification applications because of their universal approximation properties [2,3], their relation to optimal approximations [4] and classifiers [5], and the existence of useful training algorithms such as backpropagation [1]. They have the unique property that their basis functions develop during training rather than being arbitrarily chosen ahead of time. However, the MLP has long training time, is sensitive to initial weight choices, and its training error may not converge to a global minimum. Also, there is a wide gap between the MLP's theoretical capabilities and its actual performance with currently available training algorithms.

In contrast, the functional link network [6] (FLN) has a flat architecture, with pre-defined basis functions like the trigonometric functions and polynomials. The global minimum of its error function can be found by solving linear equations [6] or by techniques such as genetic algorithms [7]. As with other polynomial neural networks (PNN), fast learning rates are achieved [8] for a given network size. However, with an increase in the degree of the approximation and the number of inputs there is a combinatorial explosion in the number of basis functions used during training.

Orthonormal Least Squares has been used by Chen et al. [9] to efficiently find subsets of center vectors in radial basis functions (RBF) networks. Kamniski and Strumillo [10] used the modified Gram Schmidt orthonormalization (MGSO) to compute hidden weights in RBF networks. Surekha and Manry [12] developed the Ordered Functional Link Network (OFLN) by using MGSO to orthonormalize and order FLN basis functions according to their contribution to the decrease in the mean square error (MSE). However, it is difficult to extend OFLN training to classifiers competitive with Bayes [20] and support vector machine (SVM) [21] classifiers, when desired outputs are binary.

1.2 Thesis Organization

In this document we develop methods for calculating non-binary desired outputs for flat classification networks. Chapter 2 describes a notation for flat networks and flat network training. Chapter 3 explains the theory of non-binary desired outputs. Chapter 4 describes the first method [14] [13] for improving a classifier's desired outputs and its limitations and drawbacks. Chapter 5 describes a minimum probability of error approach and its problems. Chapter 6 explains and analyzes the final method of improving the classifier.

CHAPTER 2

FLAT NETWORK REVIEW

2.1 Mean Square Error (MSE) criterion

A flat classifier such as the FLN can be designed by minimizing the MSE function

$$E = \sum_{i=1}^{M} E(i) = \frac{1}{N_v} \sum_{i=1}^{M} \sum_{p=1}^{N_v} [t_p(i) - y_p(i)]^2 \quad (1)$$

M is the number of classes, N_v denotes the total number of training patterns and E(i) is the mean-squared error for the i^{th} class. The i^{th} desired output for the p^{th} pattern is defined as

$$t_p(i) = \begin{cases} b & \text{for } i = i_c \\ -b & \forall\ i = i_d \end{cases}$$

i_c denotes the correct class number, i_d denotes an incorrect class number and b is a positive constant such as 1. The i^{th} actual output for the p^{th} pattern is defined as

$$y_p(i) = \sum_{k=1}^{L} w(i,k) \cdot X_p(k) \quad (2)$$

where $X_p(k)$ is the k^{th} basis function for the p^{th} pattern, L is the number of basis functions and w(i,k) is the weight connecting the k^{th} basis function to the i^{th} output. The estimated class is that index i for which $y_p(i)$ is maximum. If the FLN basis functions are ordered according to their contribution to the decrease in the MSE then the network is called OFLN.

FLNs or OFLNs often have fixed number of polynomials or trigonometric basis functions. An FLN with L inputs and 2 outputs is shown in Figure 2.1

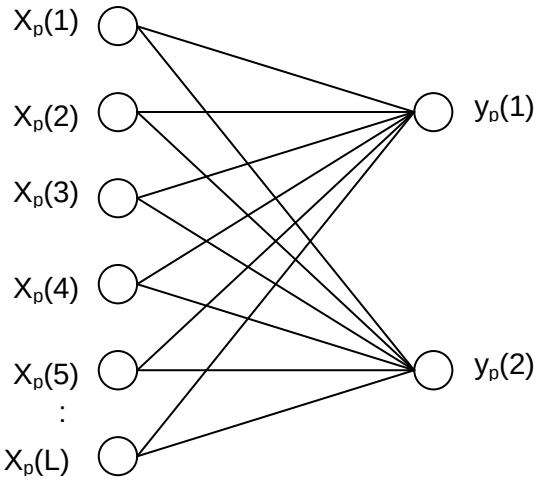

Figure 2.1 FLN structure with L inputs and 2 outputs

The minimum MSE solution for the output weight matrix **W** is found by solving

$$\mathbf{R} \cdot \mathbf{W}^{T} = \mathbf{C} \tag{3}$$

where r(i,j) is an element of the autocorrelation matrix **R,** defined as

$$r(i,j) = \frac{1}{N_v} \sum_{p=1}^{N_v} X_p(i) \cdot X_p(j) \tag{4}$$

and where element c(i,j) of the cross –correlation matrix **C** is defined as

$$c(i,j) = \frac{1}{N_v} \sum_{p=1}^{N_v} X_p(i) \cdot t_p(j) \tag{5}$$

A batch type algorithm for solving Eq. (3) is as follows:

(1) In one pass through the training data, accumulate **R** and **C**.

(2) Using Orthogonal Least Squares [11] (OLS) solve Eq. (3)

Network design through the solution of Eq. (3) is denoted as flat training. If the desired outputs are binary or -1's and 1's then the network is called as binary case in our document.

Note that OLS does not use the rows of the auto-correlation matrix(**R**) that are linearly dependent on those already processed. Effectively, OLS removes dependent basis functions $X_p(n)$ forcing **R** to be nonsingular.

2.2 Algorithm for Efficient Generation of Basis Functions

Algorithm for generating basis functions upto degree 2

Read the input vector $\mathbf{x}_p$

$X_p(1) \leftarrow 1$, $L \leftarrow 1$

For $1 \leq n \leq N$

$L \leftarrow L+1$

$X_p(L) \leftarrow x_p(n)$

$Z \leftarrow x_p(n)$

For $1 \leq m \leq n$

$L \leftarrow L+1$

$X_p(L) \leftarrow Z \cdot x_p(m)$

End

End

Algorithm for generating basis functions upto degree 3

Read the input vector $\mathbf{x}_p$

$X_p(1) \leftarrow 1$, $L \leftarrow 1$

For $1 \leq n \leq N$

$L \leftarrow L+1$

$X_p(L) \leftarrow x_p(n)$

$Z \leftarrow x_p(n)$

For $1 \leq m \leq n$

$L \leftarrow L+1$

$Z_1 \leftarrow Z \cdot x_p(m)$

$X_p(L) \leftarrow Z_1$

For $1 \leq i \leq m$

$L \leftarrow L+1$

$X_p(L) \leftarrow Z_1 \cdot x_p(i)$

End

End

End

2.3 Algorithm for Calculating Percentage Classification Error

Algorithm for calculating the Probability of Error and Percentage Classification error

(1) Initially $N_{er} = 0$, $P_e = 0$ and Percentage Classification Error = 0

(2) For $1 \leq p \leq N_v$, read the input vector ($\mathbf{X}_p$) and the correct class (i_c)

(3) Find the output vector ($\mathbf{y}_p$) using input vector ($\mathbf{X}_p$) and weight matrix (**W)**

(4) Find the index i for which $y_p(i)$ is maximum, that index i is the estimated class

(5) If estimated class = correct class then goto step(7)

(6) Increment N_{er}

(7) If the entire data file is read then goto step(8) else goto step(2)

(8) $P_e = N_{er}/N_v$, Percentage Classification error= $(100 \times N_{er})/ N_v$

CHAPTER 3

POTENTIAL OPTIMALITY OF REGRESSION

3.1 Problems with Regression Based Classifiers

Let $N_v(i)$ denote the number of patterns belonging to class i, S(i) denote the set of patterns that correspond to class i, $f_x(\mathbf{x}_p)$ denote the probability density of feature vector $\mathbf{x}_p$, $f(\mathbf{x}_p|i)$ the conditional density of feature vector $\mathbf{x}_p$ given that it belongs to class i and P(i) the probability that feature vector comes from class i.

The optimal Bayesian discriminant $d_p(i)$ for feature vector $\mathbf{x}_p$ is given as

$d_p(i) = P(i|\mathbf{x}_p)$

where $P(i|\mathbf{x}_p)$ is the aposteriori probability given that vector $\mathbf{x}_p$ belongs to class i. Let the desired output $t_p(i)$ for correct class i_c in this case be defined as

$t_p(i)= \delta(i-i_c)$

Then the expected squared error between network output y(i) and optimal Bayes discriminant is given by

$$E_B = \sum_{i=1}^{M} E[(d(i) - y(i))^2]$$

where E[.] is the Expected operator and and y(i) is a random variable storing the i^{th} actual network output. Note that $y_p(i)$ is the actual i^{th} output when the input vector is $\mathbf{x}_p$

Theorem 1: As the training patterns N_v increase, E approaches the (E_B + K), where K is a constant.

Thus E = E_B + K, it implies that network with estimated output $\mathbf{y}_p$, which minimizes the MSE yields the optimal Bayes discriminant function in the minimum mean squared error sense.

Unfortunately; Classifiers designed via regression have the following problems. They are,

1) Theorem 1 does not mention anything about the network structure.

2) Minimizing E_B is not the same as minimizing P_e even though y(i) = d(i) would minimize P_e .

3) For example, if the actual outputs are better than the desired outputs as $y_p(i_c) > t_p(i_c)$ or $y_p(i_d) < t_p(i_d)$ then the MSE(E) increases but the probability of classification error decreases.

4) The weight matrix **W** obtained by minimizing the MSE is not optimal; the weight matrix for the SVM or Bayes classifier is not obtained by minimizing the MSE.

One encounters difficulties when trying to design flat classifiers directly. Minimum Probability of Error (P_e) classifiers known as Bayes classifiers, often assume that the basis vector $\mathbf{X}_p$ is Gaussian. This assumption is clearly wrong in FLN's because the 2^{nd} degree basis functions $X_p(i)^2$ are clearly not Gaussian.

Difficulties with SVM are as follows

1) SVM's are binary classifiers (two class classifiers), inorder to do multiclass classification pair wise classifications can be used (one class against all others, for all classes) [23]
2) The high algorithmic complexity and extensive memory requirements of the required quadratic programming [24].

3.2 The potential for MSE-Based Optimal Classifiers

Let $\mathbf{W}^{opt}$ denote an output weight matrix for a classifier that is optimal in some sense, such as the minimum probability of error or Bayes classifier, or the SVM.

Theorem 2: $\mathbf{W}^{opt}$ is the solution to a least squares problem

Proof: Given $\mathbf{W}^{opt}$, the optimal cross-correlation matrix $\mathbf{C}^{opt}$ is found from Eq. (3) as

$$\mathbf{C}^{opt} = \mathbf{R} \cdot (\mathbf{W}^{opt})^{T} \tag{6}$$

From Eq. (5), the optimal cross-correlation matrix $\mathbf{C}^{opt}$ can be expressed as

$$\mathbf{C}^{opt} = \frac{1}{N_v} \sum_{\mathbf{p}=1}^{\mathbf{Nv}} \mathbf{X}_p (\mathbf{t}_p)^{\mathbf{T}} \tag{7}$$

If $\mathbf{C}^{opt}$ and $\mathbf{R}$ are known in (6), we can solve for $\mathbf{W}^{opt}$.

In designing the classifier, we assume that the basis vectors $\mathbf{X}_p$ are given and unchangeable. The only component that is under user control is the desired output vector $\mathbf{t}_p$. $\mathbf{C}^{opt}$ can be found from Eq. (7) by using good desired outputs $\mathbf{t}_p{}^{opt}$ since the basis vectors $\mathbf{X}_p$ are unchangeable. So in order to get $\mathbf{W}^{opt}$ we need to use good desired

outputs $\mathbf{t}_p^{opt}$. We can get good desired outputs $\mathbf{t}_p^{opt}$ by changing the desired outputs $\mathbf{t}_p$ appropriately.

Various classifiers with desirable properties can be designed through regression, merely through the proper choice of the desired output $\mathbf{t}_p^{opt}$.

Assume that $\mathbf{C}^{opt}$ is available and that we want to generate the desired output vectors $\mathbf{t}_p$. Rewrite (7) as

$$\begin{bmatrix} c(i,1) \\ c(i,2) \\ M \\ c(i,L) \end{bmatrix} = \begin{bmatrix} a(1,1) & a(1,2) & \Lambda & a(1,N_v) \\ a(2,1) & a(2,2) & \Lambda & a(2,N_v) \\ M & M & O & M \\ a(L,1) & a(L,2) & \Lambda & a(L,N_v) \end{bmatrix} \begin{bmatrix} t(i,1) \\ t(i,2) \\ M \\ t(i,N_v) \end{bmatrix} \tag{8}$$

where c(i,n) is an element of $\mathbf{C}^{opt}$. The n^{th} row p^{th} column element of the L by N_v matrix $\mathbf{A}$ is $X_p(n)/N_v$, and t(i,p) denotes an element of $\mathbf{t}_p$. Assuming that $L < N_v$ and that $\mathbf{A}$ has rank L, Eq. (8) is an underdetermined set of equations for the vector $t_p(i)$ and has uncountably many exact solutions. Assume that $\mathbf{C}^{opt}$ is available and that we want to generate the desired output vectors $\mathbf{t}_p$. Thus it is easy to generate desired output vectors that produce $\mathbf{C}^{opt}$.

The catch is that this must be done without pre-knowledge of $\mathbf{W}^{opt}$ or $\mathbf{C}^{opt}$. Iterative algorithms designed to find the desired outputs $\mathbf{t}_p^{opt}$ are called OUTPUT RESET (OR) algorithms and the iterations are called OR iterations.

CHAPTER 4

BASIC OUTPUT RESET (OR) ALGORITHM [ALGORITHM 1]

4.1 Basic OR Concepts

The mean square error E from Eq.(1) is

$$E = \frac{1}{N_v} \sum_{i=1}^{M} \sum_{p=1}^{N_v} z_p(i)^2 \tag{9}$$

where $z_p(i) = y_p(i) - t_p(i)$ denotes the i^{th} class residual error for the p^{th} pattern.

Lemma 1: let q denote an outlier training pattern for a neural net classifier, such that $|y_q(i)|$ is large**.** Then

$$\lim_{|y_q(i)| \to \infty} E \to \infty$$

Each instance of residual error contains at least two types of bias [14]. The first type of bias a_p is an additive constant, inherent to each output vector. The second type of bias $d_p(i)$ are error components due to individual output values having the correct sign but magnitude larger than b. We mechanize the removal of biases by introducing a new desired output vector $\mathbf{t}'_p$ which is obtained as

$t'_p(i) = t_p(i) + a_p + d_p(i)$ and $d_p(i)$ is a function of p and i.

Removal of these biases has no immediate affect on the class recognition error count.

Training error to be minimized is

$$E' = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^{M} [t'_p(i) - y_p(i)]^2 \tag{10}$$

Our goal now is to find a_p, $d_p(i)$ and $y_p(i)$, that minimize E', under the following conditions:

(1) The difference $|t'_p(i_c) - t'_p(i_d)|$ must be larger than or equal to 2b. Without this condition, E' can be minimized by letting the network weights and the difference $|t'_p(i_c)-t'_p(i_d)|$ be equal to zero.

(2) Each change made to a_p, $d_p(i)$ and $t'_p(i)$, through changes in the network weights, must reduce E' or keep it changed.

A sufficient condition for finding a_p is that the gradient of E' with respect to a_p be zero yielding

$$a_p = \frac{1}{M} \sum_{i=1}^{M} [z_p(i) - d_p(i)] \tag{11}$$

Consider the addition of a_p to each desired output $y_p(i)$. As a result, the distances between correct class residual $z_p(i_c)$ and the other $z_p(i_d)$ remain unchanged.

Therefore, the classification error count remains the same, and since a_p is specifically found to minimize E', Condition 2 is satisfied.

Values for $d_p(i)$ are found by minimizing square error term $[d_p(i) + a_p - z_p(i)]^2$, yielding $d_p(i) = z_p(i) - a_p$. But, in order to satisfy Condition 1, we must also constrain $d_p(i)$ such

that $d_p(i_c) \geq 0$ and $d_p(i_d) \leq 0$. A non-zero value for $d_p(i)$ is said to be an active bias whereas a zero value is said to be inert. Condition 1 constraints are summarized by

$$d_p(i_c) = \begin{cases} z_p(i_c) - a_p \text{ if } a_p < z_p(i_c) : \text{active} \\ 0 \qquad \text{otherwise} \quad : \text{inert} \end{cases} \tag{12}$$

$$d_p(i_d) = \begin{cases} z_p(i_d) - a_p \text{ if } a_p > z_p(i_d) : \text{active} \\ 0 \qquad \text{otherwise} \quad : \text{inert} \end{cases} \tag{13}$$

According to (12) and (13), $d_p(i)$ is active if the sign of $z_p(i)$ is correct but has magnitude larger than a_p.

4.2 Iterative Solution for Algorithm 1

It is not obvious how to simultaneously find the exact values for a_p and $d_p(i)$. An approximation method for finding $t'_p(i)$ iterates through (11)-(13) until converging on a stable estimate of a_p. An OR flat training algorithm denoted as Algorithm 1 is as follows:

(1) for each iteration i_t, where $1 \leq i_t \leq N_{it}$

(2) for each vector $\mathbf{X}_p$, read the correct class(i_c) until $1 \leq p \leq N_v$

a) Calculate the networks outputs $y_p(i)$ using current network weights.

b) Find $t'_p(i)$ by iterating through equations (11)-(13) until no change in a_p or a fixed number of iterations N_{itr} is reached.

c) Accumulate the auto- and cross-correlations needed in the flat training, where $t'_p(i)$ replaces $t_p(i)$.

(3) Find the output weights w(i,j) using flat training. Go back to the step (2) for another iteration, if necessary.

Given $\mathbf{X}_p$, $t_p(i)$ a closed form OR for $t'_p(i)$ can be found from [13].

4.3 Convergence of Basic algorithm (algorithm1)

Theorem 3: The weights converge in every iteration.

proof:

Let $\mathbf{W}$ denote the weight matrix for the current OR iteration and Let $\mathbf{W}'$ denote the weight matrix for the previous OR iteration.

1) In each OR iteration, we do the following

$\mathbf{W} \leftarrow \mathbf{W}'$

If $z_p(i) > a_p$ and $i = i_c(p)$, then $p \in S_c(i)$ and $t'_p(i) \leftarrow y_p(i)$.

If $z_p(i) < a_p$ and $i \neq i_c(p)$, then $p \in S_c(i)$ and $t'_p(i) \leftarrow y_p(i)$.

Else $t'_p(i) \leftarrow t_p(i) + a_p + d_p(i)$ and $p \in S_e(i)$.

Finding the new desired outputs $t'_p(i)$ reduces $E'(i)$ which can be rewritten using Eq.(10) as

$$E'(i) = \frac{1}{N_v}\sum_{p=1}^{N_v}[t'_p(i) - y_p(i)]^2 = E_c(i) + E_e(i) \qquad (14)$$

$$E_c(i) = \sum_{p \in S_c(i)} [\sum_{n=1}^{L} w(i,n)X_p(n) - \sum_{n=1}^{L} w'(i,n)X_p(n)]^2,$$

$$E_e(i) = \sum_{p \in S_e(i)} [t'_p(i) - \sum_{n=1}^{L} w'(i,n)X_p(n)]^2$$

2) Solving linear equations in Eq. (14) we get **W**

$$\mathbf{R} \cdot \mathbf{W}^{T} = \mathbf{C} \tag{15}$$

Comments

(1) In each step (1) of section (4.3) $E'(i) = E_e(i)$ since $E_c(i) = 0$.

(2) $E'(i)$ converges, since it forms a sequence of non-increasing nonnegative numbers.

(3) The convergence of $E'(i)$ implies that its subsequence $E_e(i)$ also converges.

(4) As $E'(i)$ and $E_e(i)$ converge to the same limit, $E_c(i)$ converges to zero in each step (2) of section (4.3), meaning that

$$\mathbf{e}^{T}\mathbf{R}_{c}(i)\mathbf{e} \to 0$$

where the nx1 element of vector **e** is $e(n) = w'(i,n) - w(i,n)$

(5) For positive definite $\mathbf{R}_c(i)$,

$$\mathbf{W}' - \mathbf{W} \to 0 \tag{16}$$

Equating $\partial E'(i)/\partial w(i,k)$ to zero and using

$$\mathbf{R}_{c}(i) = \sum_{p \in \mathbf{S}_{c}(\mathbf{i})} \mathbf{X}_{p} \cdot \mathbf{X}_{p}^{T},$$

$$\mathbf{R}_{e}(i) = \sum_{p \in S_{e}(i)} \mathbf{X}_{p} \cdot \mathbf{X}_{p}^{T},$$

$$\mathbf{C}_{e}(i) = \sum_{p \in S_{e}(i)} \mathbf{X}_{p} \cdot (\mathbf{t}'_{p}(i))^{\backslash T}$$

we get

$$\mathbf{R}(\mathbf{W}(i))^{\mathbf{T}} = \mathbf{R}_{c}(i) \cdot (\mathbf{W}'(i))^{\mathbf{T}} + \mathbf{C}_{e}(i) = \mathbf{R}(\mathbf{W}')^{\mathbf{T}} + [\mathbf{C}_{e}(i) - \mathbf{R}_{e}(i) \cdot (\mathbf{W}'(i))^{\mathbf{T}}] \tag{17}$$

and

$$(\mathbf{W}(i))^{\mathbf{T}} = (\mathbf{W}'(i))^{\mathbf{T}} + \mathbf{R}^{-1} \cdot [\mathbf{C_e}(i) - \mathbf{R}_e(i) \cdot (\mathbf{W}'(i))^{\mathbf{T}}] \tag{18}$$

As $\mathbf{W}^{\mathrm{T}} - \mathbf{W'}^{\mathrm{T}} \rightarrow \mathbf{0}$ according to (16),

$$\mathbf{R}_e \mathbf{W}^{\mathbf{T}} = \mathbf{C}_e \tag{19}$$

Since there are a finite number of matrices $\mathbf{C}_e$ and $\mathbf{R}_e$, $\mathbf{W}$ converges.

4.4 Problems

Here we analyze the sensitivity of Algorithm 1 to outliers.

Algorithm 1 is not equivalent to the minimum probability of error approach as it is sensitive to misclassified outliers.

Lemma 2**:** For a neural net classifier, trained using E′,

$$\lim_{y_q(i_c) \rightarrow -\infty} E' \rightarrow \infty$$

$$\lim_{y_q(i_d) \rightarrow +\infty} E' \rightarrow \infty$$

If an outlier is correctly classified then Algorithm 1 contributes a small mean square error from that particular pattern to the final mean square error which is not a problem, Let E_o denote the normal value of E′ which is a constant, not counting the effect of outliers for patterns which are incorrectly classified. Then

$$\lim_{y_q(i_c) \rightarrow +\infty} E' \rightarrow E_o$$

$$\lim_{y_q(i_d) \rightarrow -\infty} E' \rightarrow E_o$$

If an outlier is incorrectly classified then Algorithm 1 will contribute a large mean square error from that particular pattern to the final mean square error which could change the weights and result in a very bad classifier, so this needs to be avoided.

CHAPTER 5

OR FOR MINIMUM PROBABILITY OF ERROR [ALGORITHM 2]

In this chapter we derive new version of the OR algorithm that solves the problems described in section 4.2

5.1 Algorithm 2

The goal in this subsection is to develop an OR algorithm in which the MSE is proportional to the probability of error ($E' \sim P_e$). In order to do this, we define $\mathbf{t}'_p$ as follows:

Case 1: If pattern p is correctly classified, $t'_p(i) = y_p(i)$ for all i, so this pattern does not contribute to E′.

Case 2: If pattern p is incorrectly classified and $y_p(i_d) \geq y_p(i_c)$ for K values of i_d, then

$$t'_p(i_c) = y_p(i_c) + \varepsilon,$$

$$t'_p(i_d(k)) = y_p(i_d(k)) - \varepsilon \text{ for } 1 \leq k \leq K \text{ where}$$

$$\varepsilon = \sqrt{\frac{2}{K+1}} \cdot b$$

Else,

$$t'_p(i) = y_p(i)$$

Note that this pattern contributes $2b^2$ to E′, before the division by N_v.

We can summarize cases (1) and (2) of this section as

$$d_p(i) = \begin{cases} 0 & : \text{if correctly classified or if } i = i_d \text{ and } y_p(i) < y_p(i_c) \\ -\sqrt{\dfrac{2}{K+1}} \cdot b & : \text{if incorrectly classified, } i \neq i_c \text{ and } y_p(i) \geq y_p(i_c) \\ \sqrt{\dfrac{2}{K+1}} \cdot b & : \text{if incorrectly classified and } i = i_c \end{cases} \quad (20)$$

Using $d_p(i)$ from Eq. (20) in $t'_p(i)$ we get

$$t'_p(i) = y_p(i) + d_p(i) \quad (21)$$

So every incorrectly classified pattern contributes the same value, $2b^2$, to E' before the division by N_v.

Lemma 3: $E' = 2b^2 \cdot P_e$

Each misclassified pattern adds $2b^2$ to the cumulative squared error, before the division by N_v. If there are N_{er} such patterns, the cumulative error is $2b^2 \cdot N_{er}$. Dividing by N_v we get

$$2b^2 \cdot N_{er} / N_v = 2b^2 \cdot P_e$$

5.2 Iterative solution for Algorithm2

The iterative flat training algorithm in this case is as follows:

(1) for each iteration i_t , where $1 \le i_t \le N_{it}$

(2) for each vector $\mathbf{X}_p$, read the correct class(i_c) until $1 \le p \le N_v$

a) Calculate the networks outputs $y_p(i)$ using current network weights and increment i_t .

b) Find $t'_p(i)$ by using Eq. (20) and Eq. (21).

c) Accumulate the auto- and cross-correlations needed in flat training, where $t'_p(i)$ replaces $t_p(i)$.

(3). Find the output weight matrix **W** using flat training. Go back to the step (2) for another iteration, if necessary.

5.3 Properties of Algorithm 2

In this subsection we show that the weight matrix keeps on changing as long as $P_e \neq 0$ which is again illustrated in Figure 5.3.

Lemma 4: The maximum contribution of an outlier pattern to the cumulative squared error is $2b^2$.

Therefore algorithm 2 is fairly immune to outliers

Theorem 4: If all patterns are correctly classified during a training iteration, training stops and the final weight matrix **W** has been found.

Proof:

Case (1): **R** is nonsingular

If all the patterns are correctly classified then $t'_p(i) = y_p(i)$ as in case(1) of section(5.1). Letting w′(i,n) designate weights which were found in the previous iteration,

$$\frac{\partial E'(i)}{\partial w(i,k)} = \frac{-2}{N_v}\sum_{p=1}^{N_v}[t'_p(i) - \sum_{n=1}^{L} w(i,n)X_p(n)]X_p(k) = 0$$

$$= \frac{-2}{N_v}\sum_{p=1}^{N_v}[\sum_{n=1}^{L} w'(i,n)X_p(n) - \sum_{n=1}^{L} w(i,n)X_p(n)]X_p(k)\,,$$

which leads to

$$\mathbf{R}\cdot(\mathbf{W}-\mathbf{W}')^T = 0 \qquad (22)$$

The solution to this homogeneous equation is

$\mathbf{W} = \mathbf{W}'$

meaning that the weights found in the current iteration equal those from the last iteration.

Case (2): **R** is singular

OLS in section (2.1) removes the linearly dependent basis functions and decrements L until **R** is nonsingular. Then case (1) in Theorem 4 applies.

Corollary: In each iteration where the cross-correlation matrix equation $\mathbf{C}_d$,

$$\mathbf{C}_d = \frac{1}{N_v}\sum_{p=1}^{N_v}\mathbf{X}_p\mathbf{d}_p{}^T$$

is not a matrix of zeros, **W** will change

Proof: Letting $t'_p(i) = y_p(i) + d_p(i)$ as in Eq. (21),

$$\frac{\partial E'(i)}{\partial w(i,k)} = \frac{-2}{N_v}\sum_{p=1}^{N_v}[t'_p(i) - \sum_{n=1}^{L} w(i,n)X_p(n)]X_p(k) = 0$$

$$= \frac{-2}{N_v}\sum_{p=1}^{N_v}[d_p(i) + \sum_{n=1}^{L} w'(i,n)X_p(n) - \sum_{n=1}^{L} w(i,n)X_p(n)]X_p(k)$$

which leads to

$$\mathbf{R}\cdot(\mathbf{W} - \mathbf{W}')^T = \mathbf{C}_d\,, \tag{23}$$

Since there are a finite number of basis vectors $\mathbf{X}_p$ and vectors $\mathbf{d}_p$, there are a finite number of $\mathbf{C}_d$ matrices. For misclassified patterns $\mathbf{d}_p$ has some non zero elements

$$\mathbf{W^T} = \mathbf{W'^T} + \mathbf{R^{-1}C_d} \tag{24}$$

Since **R** is nonsingular, $\mathbf{R}^{-1}\,\mathbf{C}_d$ is not a matrix of zeros, and **W** changes in every iteration for which P_e is nonzero.

From the corollary of theorem 4, Algorithm 2 may not halt as long as $P_e > 0$. As an example, consider Figure 5.3 a first degree OFLN classifier trained using Algorithm 2. Since $P_e > 0$, training does not halt.

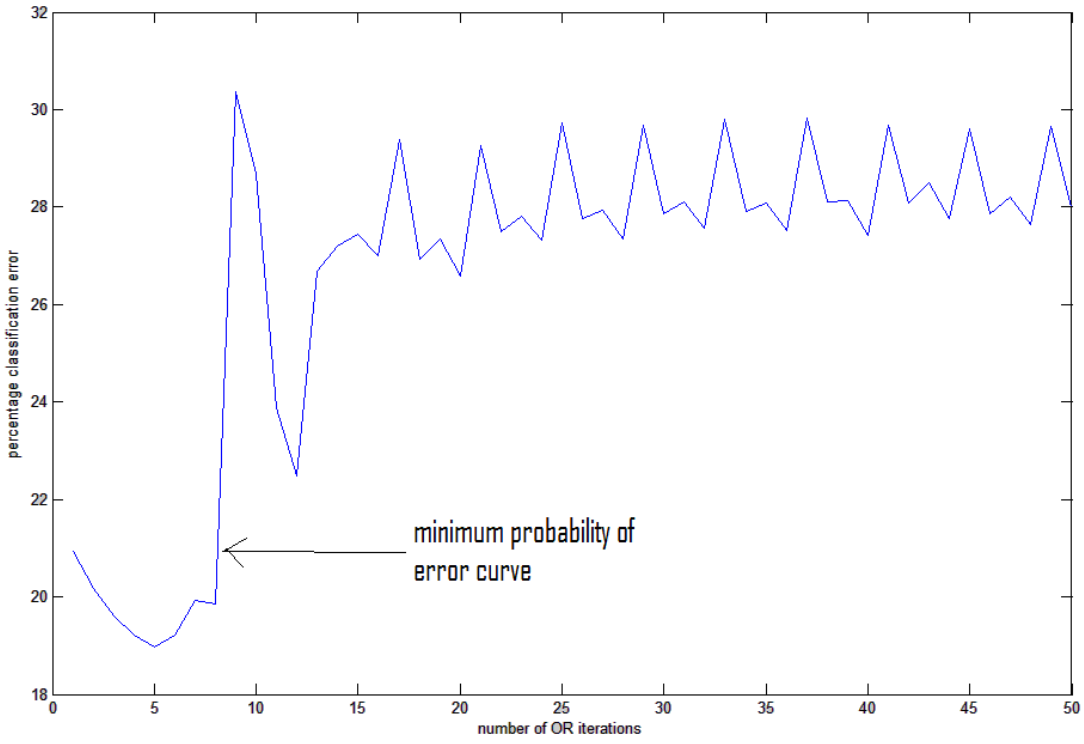

Figure 5.3 Minimum Probability of Error Curve

Some remaining problems with OR are as follows.

(1) The weight matrix **W** and P_e are not guaranteed to converge as shown in the corollary to theorem 4.

(2) Correctly classified patterns can become misclassified as the training progresses.

(3) If a pattern is misclassified then the sum of the changes made to the desired outputs is not zero, i.e.

$$\sum_{i=1}^{M}[t'_p(i) - y_p(i)] \neq 0$$

CHAPTER 6

FINAL OR METHOD [ALGORITHM 3]

In this section we attack problems (2) and (3) of algorithm 2, leading to algorithm 3.

6.1 Algorithm 3

One way to attack problem (2) is that for outputs which satisfy $y_p(i_d) < y_p(i_c)$ should also contribute to the MSE .

Case (1): If a pattern p is correctly classified then

$$d_p(i) = \begin{cases} -e^{-\frac{a}{2}[y_p(i_c)-y_p(i)]} & : \text{if} \quad \dfrac{y_p(i_c)-y_p(i)}{2} < 1 \\ 0 & : \text{if} \quad \dfrac{y_p(i_c)-y_p(i)}{2} \geq 1 \end{cases} \tag{25}$$

Using $d_p(i)$ we get

$$t'_p(i) = y_p(i) + d_p(i) \quad \text{if} \quad i = i_d$$

$$t'_p(i_c) = y_p(i_c) + d_p(i_c) \quad \text{where} \quad d_p(i_c) = \sum_{\substack{i=1 \\ i \neq i_c}}^{M} [-d_p(i)]$$

Since we are introducing the terms $d_p(i)$'s, correctly classified patterns also contribute to the final MSE so this solves problem (2).

Since

$$\sum_{i=1}^{M} d_p(i) = 0$$

this solves problem (3) for correctly classified patterns.

Case (2): if a pattern is incorrectly classified then

For incorrect classes (i_d's) we do the following

- if $y_p(i_d) < y_p(i_c)$

$$t'_p(i_d) = y_p(i_d) + d_p(i_d) \text{ where } d_p(i_d) = \begin{cases} -e^{-\frac{a}{2}[y_p(i_c) - y_p(i_d)]} & \text{if } \dfrac{y_p(i_c) - y(i_d)}{2} < 1 \\ 0 & \text{if } \dfrac{y_p(i_c) - y(i_d)}{2} \geq 1 \end{cases} \quad (26)$$

- else

$$t'_p(i_d) = y_p(i_d) - \varepsilon \quad \text{where} \quad \varepsilon = \sqrt{\frac{1}{K(K+1)}} \cdot b$$

For correct class (i_c)

$$t'_p(i_c) = y_p(i_c) + K \cdot \varepsilon$$

Since we are including $d_p(i_d)$'s this solves problem (2), we are adding $K \cdot \varepsilon$ to $y_p(i_c)$ and subtracting ε from K $y_p(i_d)$'s so the sum of changes to the actual outputs are almost zero as $d_p(i_d)$'s are negligible which solves problem (3).

6.2 Iterative solution for Algorithm 3

The iterative flat training algorithm in this case is as follows:

1. for each iteration i_t , where $1 \le i_t \le N_{it}$

2. for each vector $\mathbf{X}_p$, read the correct class(i_c) until $1 \le p \le N_v$

   a) Calculate the networks outputs $y_p(i)$ using current network weights and increment i_t .

   b) Find $t'_p(i)$ by using either cases (1) or (2) depending on whether the pattern is correctly classified or incorrectly classified.

   c) Accumulate the auto- and cross-correlations needed in flat training, where $t'_p(i)$ replaces $t_p(i)$.

3. Find the output weight matrix **W** using flat training. Go back to the step (2) for another iteration, if necessary.

6.3 Analysis of Algorithm 3

Here we are analyzing the effects of $d_p(i)$ on the convergence of the weight matrix

Condition 1: $y_p(i) < y_p(i_c)$ if $i=i_d$ then $y_p(i) \in S_c(i)$

Condition 2: $y_p(i) \ge y_p(i_c)$ if $i=i_d$ then $y_p(i) \in S_d(i)$

If a pattern is correctly classified then $y_p(i_c) \in S_c(i_c)$. If the pattern is incorrectly classified then $y_p(i_c) \in S_d(i_d)$.

We have

$$E'(i) = \frac{1}{N_v}\sum_{p=1}^{N_v}[t'_p(i) - y_p(i)]^2 = \frac{1}{N_v}\sum_{y_p(i)\in S_c(i)}[\sum_{n=1}^{L} w'(i,n)X_p(n) + d_p(i) - \sum_{j=1}^{L} w(i,j)X_p(j)]^2 +$$

$$\frac{1}{N_v}\sum_{y_p(i)\in S_d(i)}[\sum_{n=1}^{L} w'(i,n)X_p(n) + \alpha_p(i) - \sum_{j=1}^{L} w(i,j)X_p(j)]^2$$

Where $\alpha_p(i) = \begin{cases} K\sqrt{\frac{1}{K(K+1)}}\cdot b & \text{if} \quad i = i_c \\ -\sqrt{\frac{1}{K(K+1)}}\cdot b & \text{if } i = i_d \end{cases}$

Now

$$\frac{\partial E'(i)}{\partial w(i,k)} = \sum_{n=1}^{L} w'(i,n)R(n,k) + \frac{1}{N_v}\sum_{y_p(i)\in S_c(i)} d_p(i)X_p(k) - \sum_{j=1}^{L} w(i,j)R(j,k)$$

$$+\frac{1}{N_v}\sum_{y_p(i)\in S_d(i)} \alpha_p(i)X_p(k)$$

So

$$[\sum_{n=1}^{L}[w'(i,n) - w(i,n)]R(n,k)] + \theta_c + \theta_e = 0$$

where

$$\theta_c = \frac{1}{N_v}\sum_{p\in S_c(i)} d_p(i)X_p(k) \qquad (27)$$

and

$$\theta_e = \frac{1}{N_v} \sum_{p \in S_d(i)} \alpha_p(i) X_p(k) \tag{28}$$

So,

$$\sum_{n=1}^{L} [w'(i,n) - w(i,n)] R(n,k) = -(\theta_c + \theta_e) \tag{29}$$

Here **W** denotes the weight matrix for the current OR iteration and **W′** denotes the weight matrix for the previous OR iteration. The necessary conditions for the weights to converge is $\theta_c = -\theta_e$.

Case 1) If we select $d_p(i)$ to be constant for all p and i, then in general $|\theta_c| >> |\theta_e|$ since all the outputs which satisfy condition 1 will be contributing to the final MSE and so the weights never converge in Eq. (29) even if we select a large number of OR iterations.

Case 2) if we select $d_p(i)$ to be small then in general $|\theta_e| >> |\theta_c|$ and so the weights never converge as proved in the corollary of theorem 4.

We need to select optimal $d_p(i)$ and this can be done by making $d_p(i)$ as a function of $[y_p(i_c)-y_p(i_d)]$.

If we select $d_p(i)$ as an increasing function of $[y_p(i_c) - y_p(i_d)]$ then as $y_p(i_c)$ and $y_p(i_d)$ become better we will be subtracting more and adding more so in this case if a classifier is very good then we will changing the weights which will never converge. So $d_p(i)$ should be a decreasing function of $[y_p(i_c) - y_p(i_d)]$ and the slope of the curve $[y_p(i_c) - y_p(i_d)]$ shouldn't decrease too rapidly nor it should remain flat. It is possible to use decreasing ramp or decreasing exponential.

It is possible that $\theta_c = -\theta_e$ after some OR iterations so that the weights converge and the percentage classification error remains constant.

CHAPTER 7

NUMERICAL RESULTS

Percentage classification errors of Algorithm 3, Algorithm1 and Bayes Gaussian classifier for both training data files and validation data files are displayed below and brief description about the files are also given. For Algorithm 3, a=7 is used in Eq. (25) and Eq. (26) and 10 OR iterations are used for all the degrees. The description of the Bayes Gaussian classifier used here is given in appendix B. 10-fold validation is done in all the cases.

In the table the minimum percentage classification error is shown for the network.

For all the plots shown below, y-axis is the percentage classification error and x-axis is the number of basis functions.

7.1 Gong data file

The raw data consists of images from hand printed numerals collected from 6,000 people by the Internal Revenue Service. We randomly chose 600 characters from each class to generate 6,000 character training data. Images are 32 by 24 binary matrices. An image scaling algorithm is used to remove size variation in characters. The feature set contains 16 elements. The 10 classes correspond to 10 Arabic numeral [16].

Total number of patterns=6000

Training patterns=5400, Validation patterns=600.

Inputs=16, Classes=10.

Table 7.1 Comparison of Percentage Classification Error for Gong data file.

	Training			Validation		
	Degree 1	Degree 2	Degree 3	Degree 1	Degree 2	Degree 3
Algorithm 3	8.986874	4.888149	3.150305	9.38436	7.004992	6.905158
Algorithm 1	9.755962	5.416898	3.734516	10	7.271215	7.004992
Bayes Gaussian Classifier	12.78055	7.709373	5.784803	12.89517	8.43594	7.836938
Binary case	13.02274	7.676095	5.794047	13.22795	8.402662	7.836938

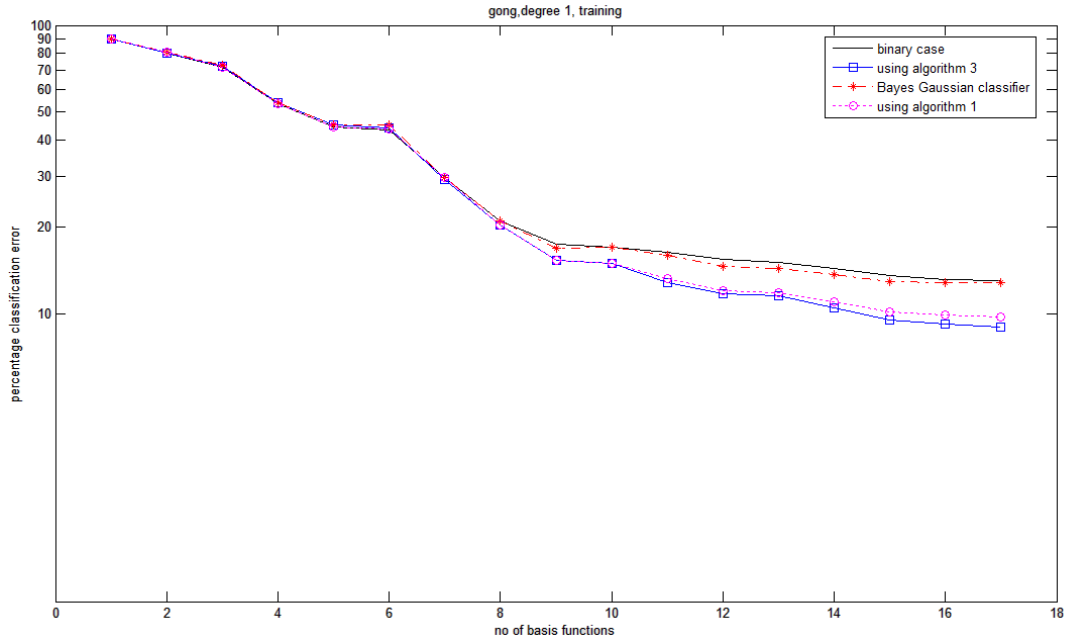

Figure 7.1 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 1.

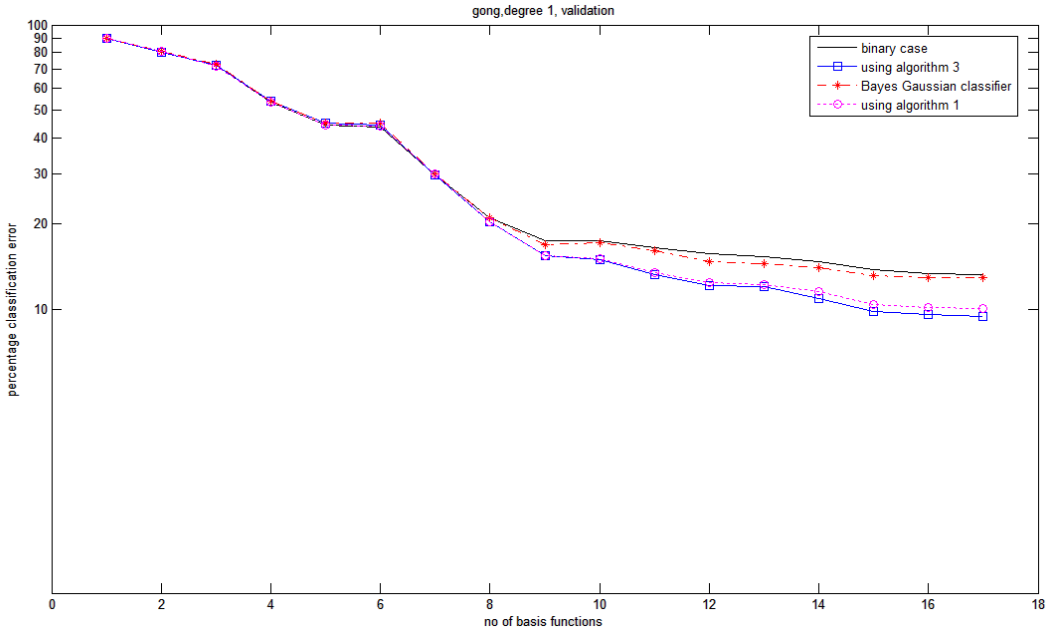

Figure 7.2 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 1.

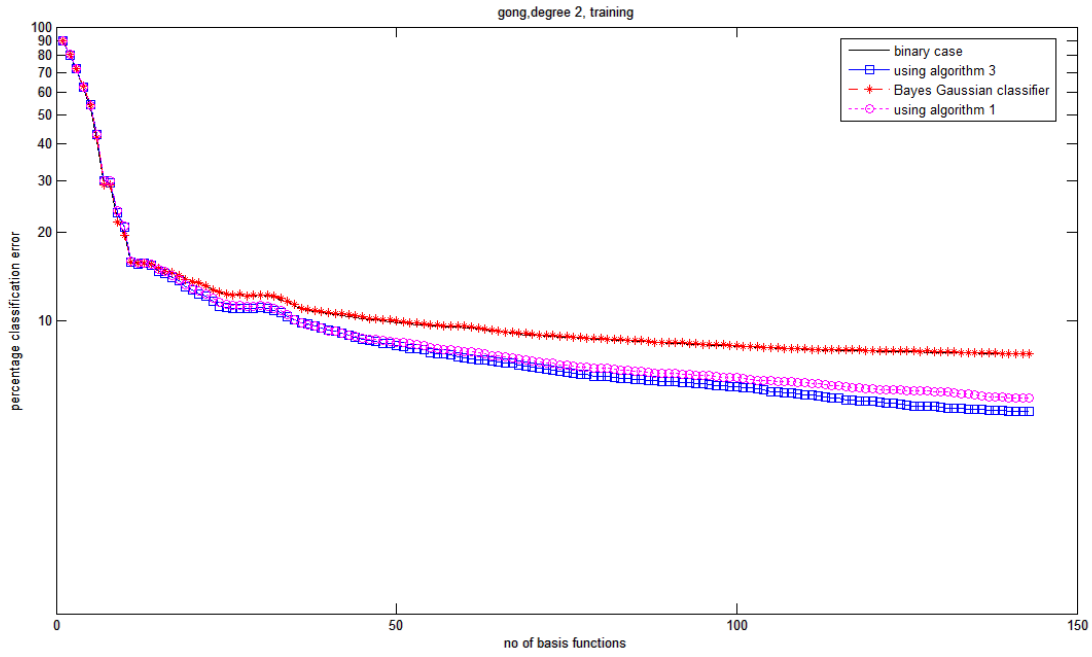

Figure 7.3 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 2.

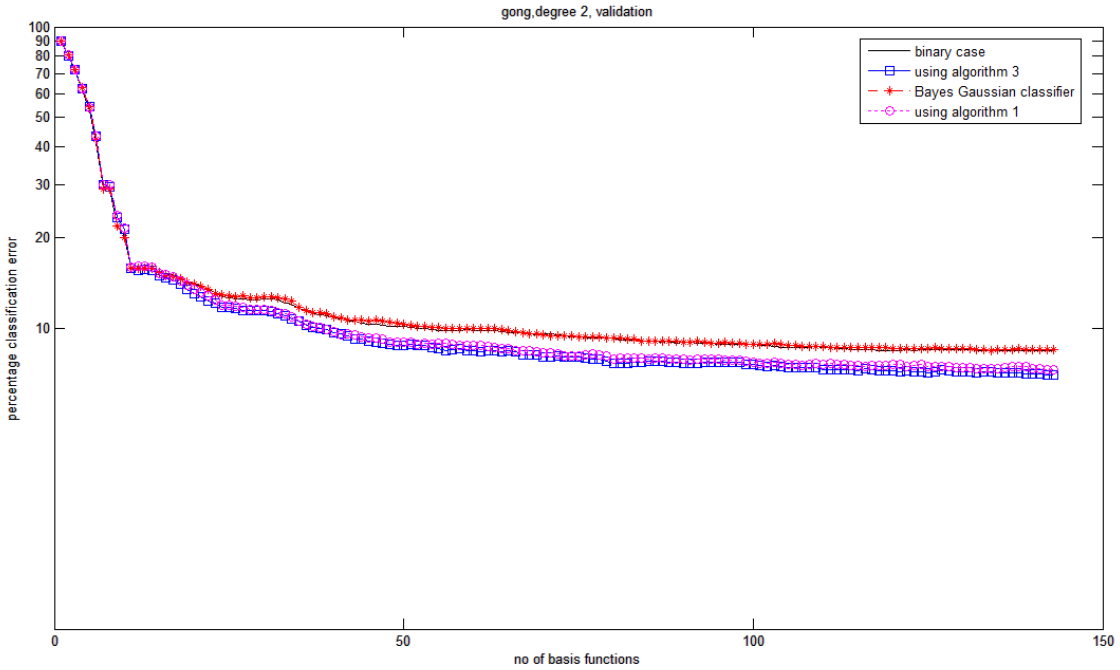

Figure 7.4 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 2.

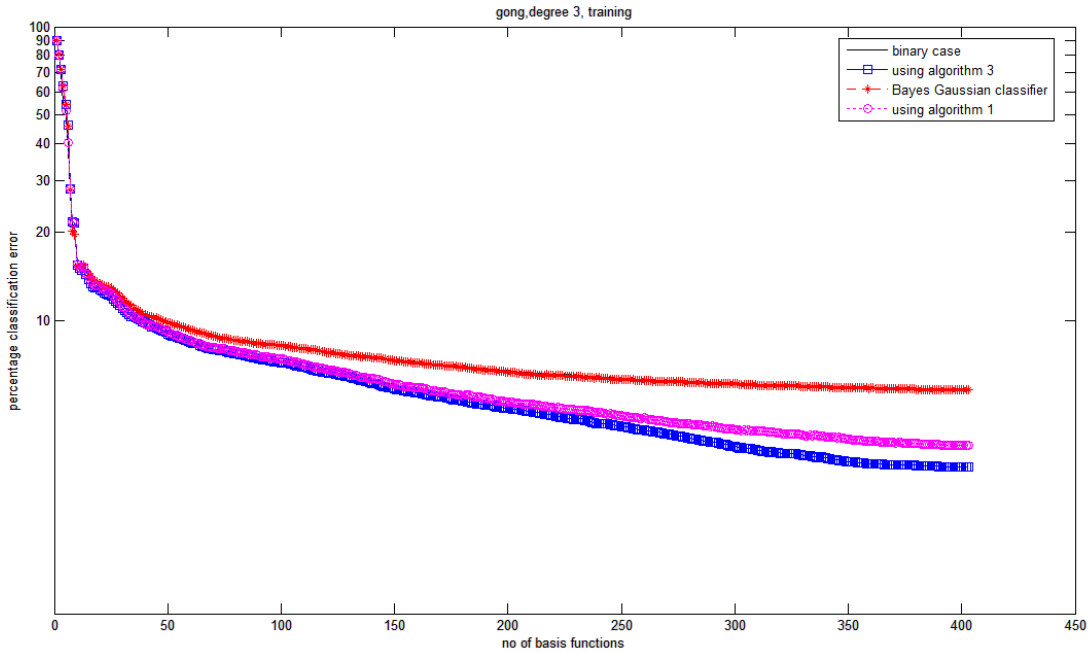

Figure 7.5 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 3.

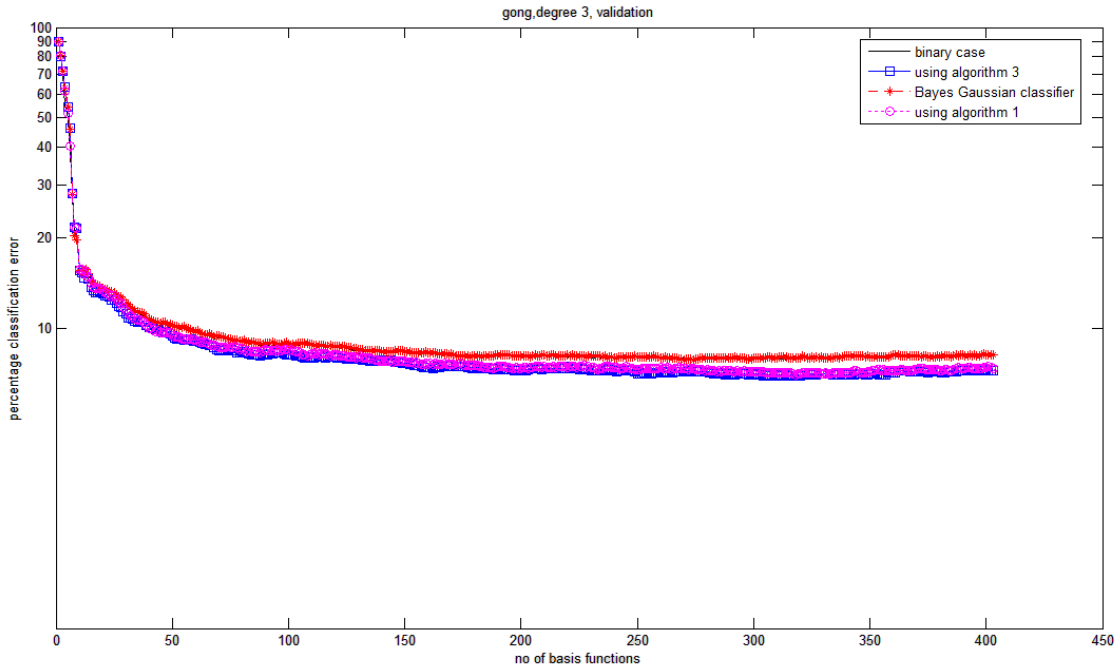

Figure 7.6 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 3.

7.2 Comf18 Data file

This training data file is generated segmented images. Each segmented region is separately histogram equalized to 20 levels. Then the joint probability density of pairs of pixels separated by a given distance and a given direction is estimated. We use 0, 90, 180 and 270 degrees for the directions and 1, 3 and 5 pixels for the separations. The density estimates are computed for each classification window. For each separation, the co-occurrences for the four directions are folded together to form a triangular matrix. From each of the resulting three matrices, six features are computed: angular second moment, contrast, entropy, correlation, and the sums of the main diagonal. This results

in 18 features for each classification window [17]. Comf18 data file is split into two files one for training and another for testing.

Total number of patterns=12392

Training patterns=11153, Validation Patterns=12392

Inputs=18, Classes=4.

Table 7.2 Comparison of Percentage Classification Error for Comf18 data file

	Training			Validation		
	Degree 1	Degree 2	Degree 3	Degree 1	Degree 2	Degree 3
Algorithm 3	18.68639	12.89652	10.64617	26.68146	20.26854	19.73305
Algorithm 1	19.55715	13.5908	11.46676	27.04952	20.52704	19.67733
Bayes Gaussian Classifier	22.73384	15.38874	13.37938	29.96258	23.37269	22.11285
Binary case	21.84336	14.95337	12.97267	29.27139	21.63222	20.59228

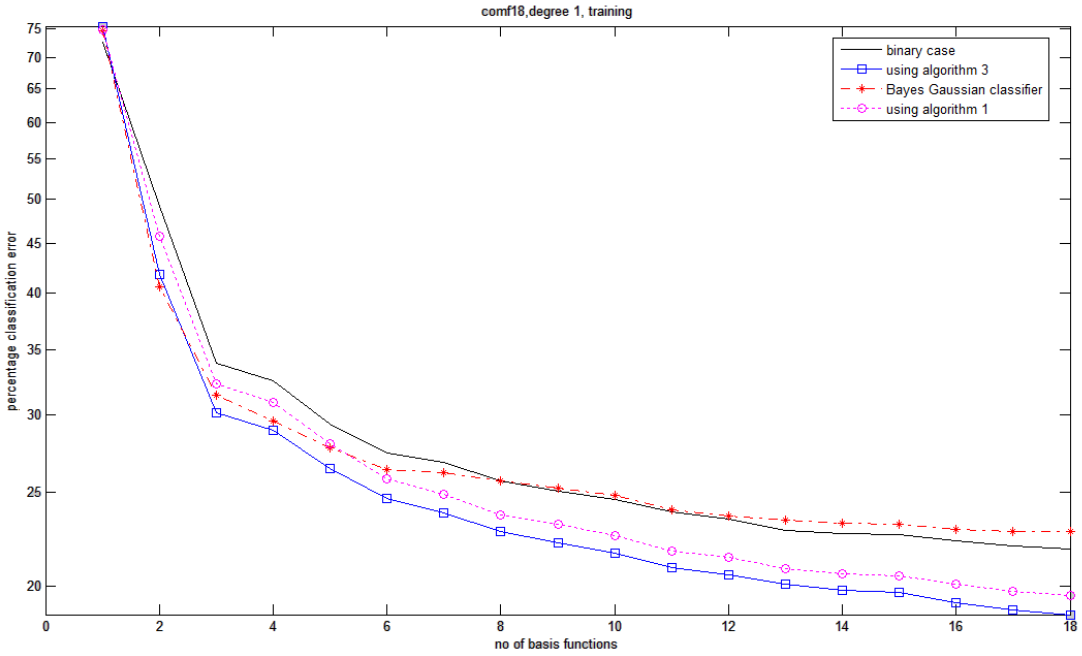

Figure 7.7 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 1.

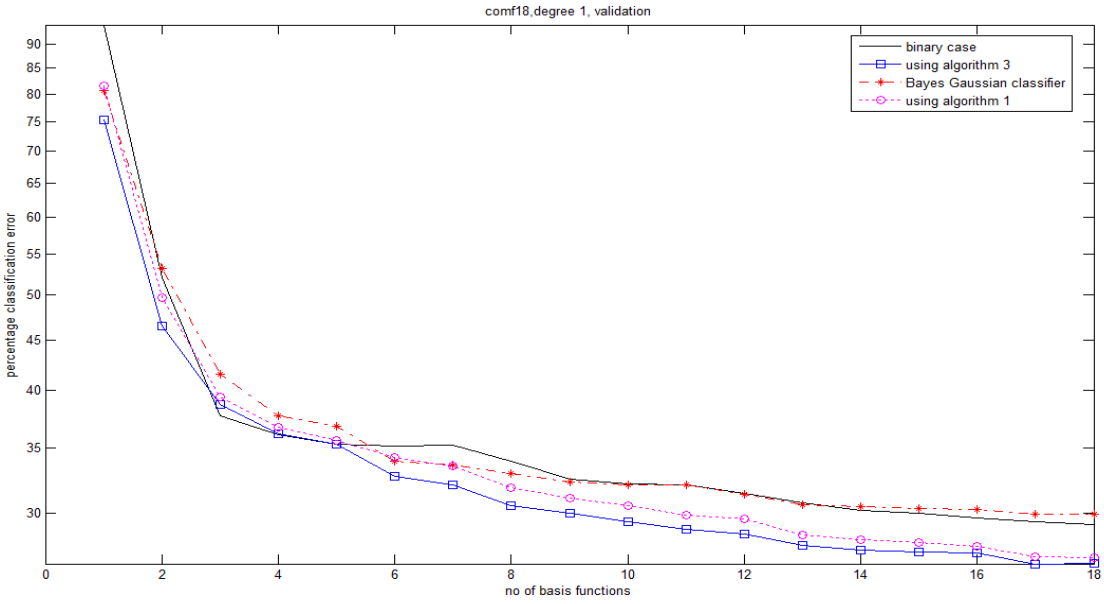

Figure 7.8 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 1.

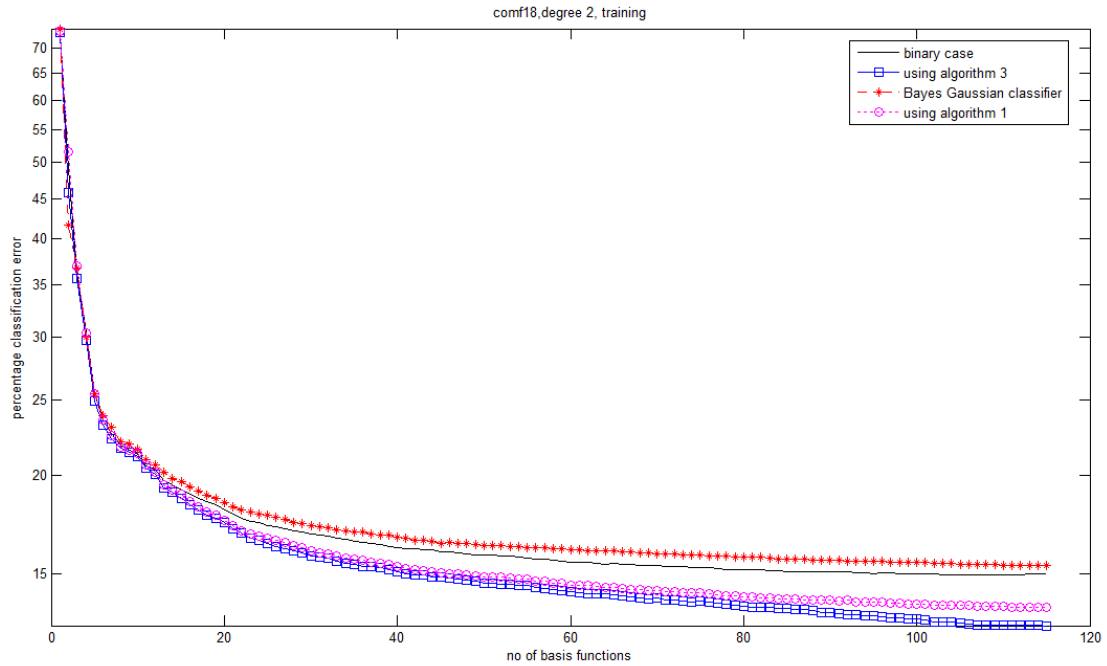

Figure 7.9 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 2.

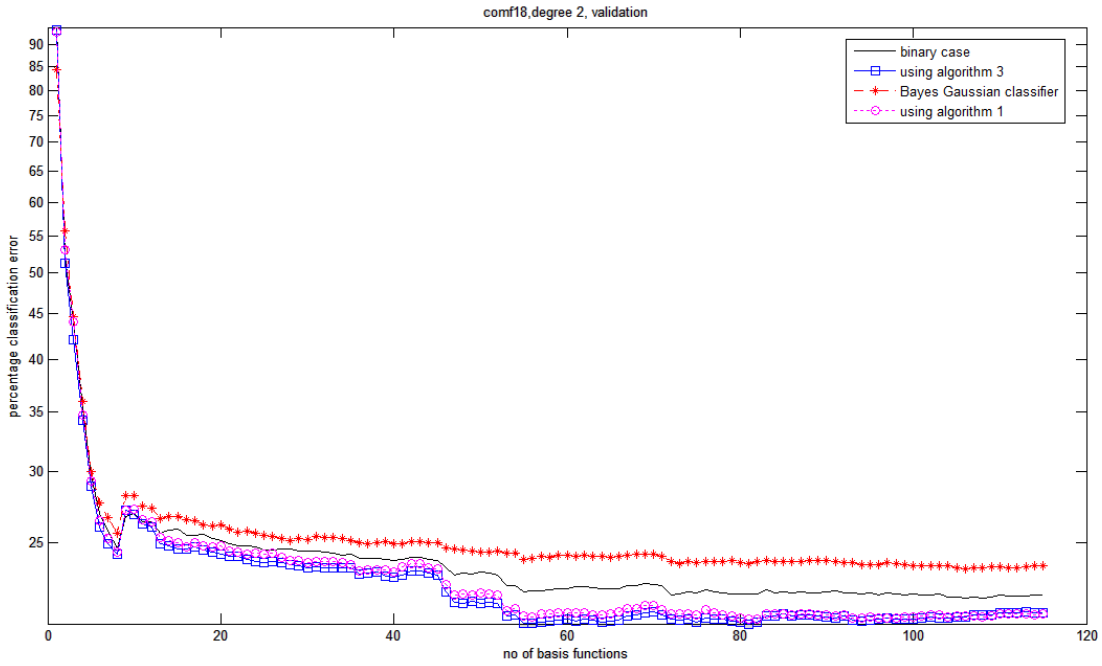

Figure 7.10 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 2.

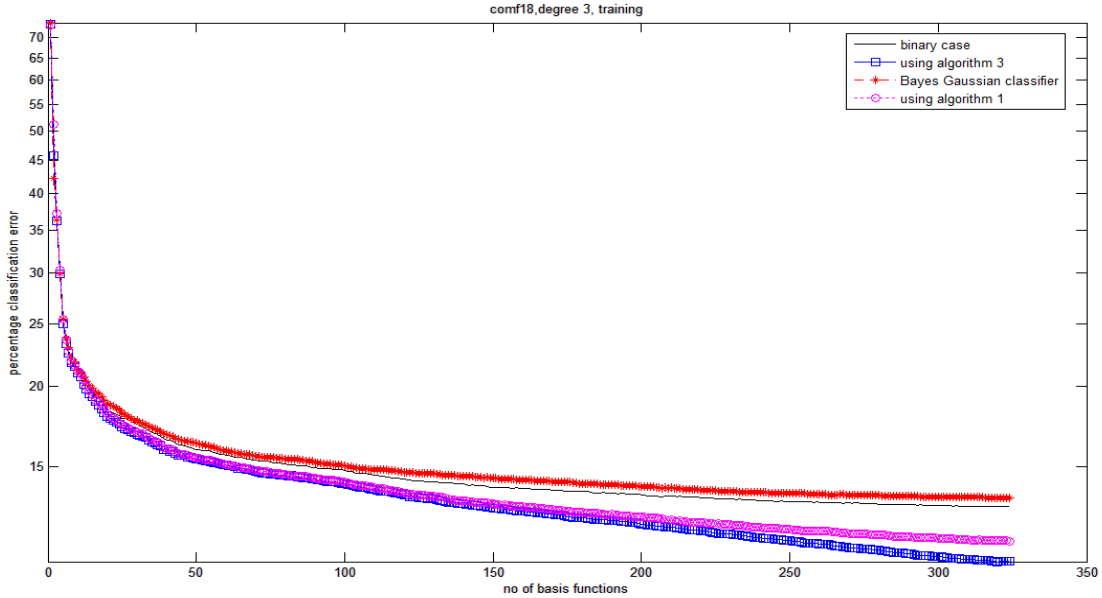

Figure 7.11 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 3.

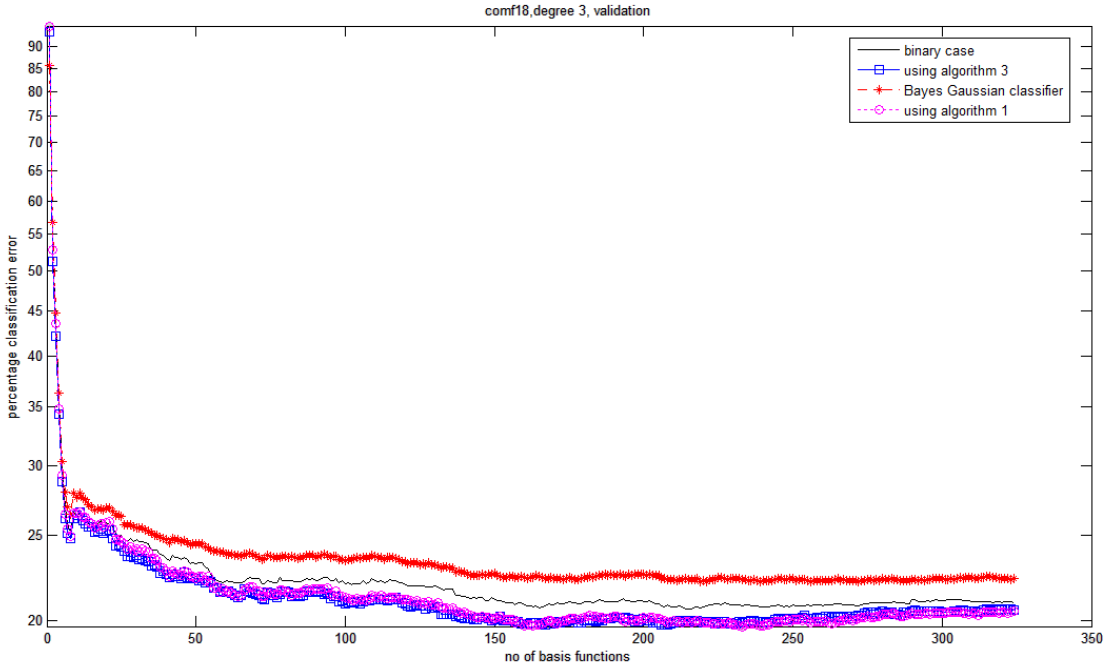

Figure 7.12 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 3.

7.3 Grng Data File

This file is a geometric shape recognition data file consists of four geometric shapes, ellipse, triangle, quadrilateral, and pentagon. Each shape consists of a matrix of size 64x64. For each shape, 200 triangle patterns were generated using different degrees of deformation. The deformations include rotation, scaling, translation and oblique distortion. The feature set is ring-wedge energy (RNG), and it has 16 features [15].Grng data file is split into two files one for training and another for testing.

Total number of patterns=800

Training patterns=720, Validation patterns=80

Inputs=16, Classes=4.

Table 7.3 Comparison of Percentage Classification Error for Grng data file

	Training			Validation		
	Degree 1	Degree 2	Degree 3	Degree 1	Degree 2	Degree 3
Algorithm 3	10.21948	0.233196	0	14.93827	5.308642	5.185185
Algorithm 1	11.82442	0.823045	0	14.69136	5.185185	5.679013
Bayes Gaussian Classifier	15.56927	2.098765	0.027435	18.51852	5.925926	5.185185
Binary case	15.4321	2.002743	0.027435	19.1358	5.925926	5.061729

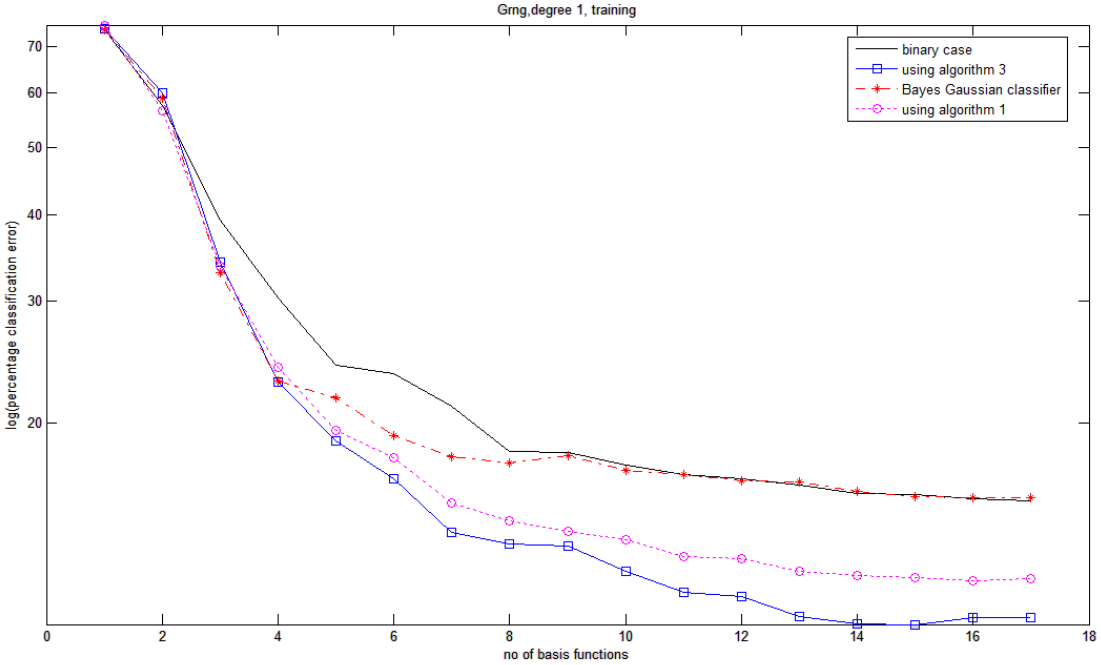

Figure 7.13 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 1.

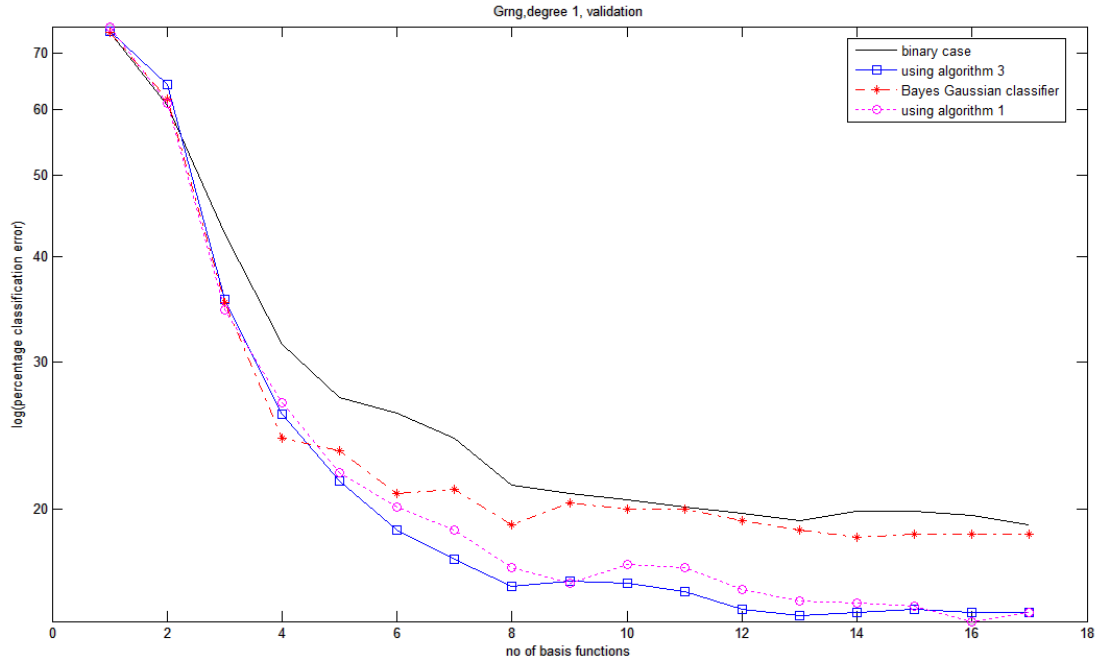

Figure 7.14 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 1.

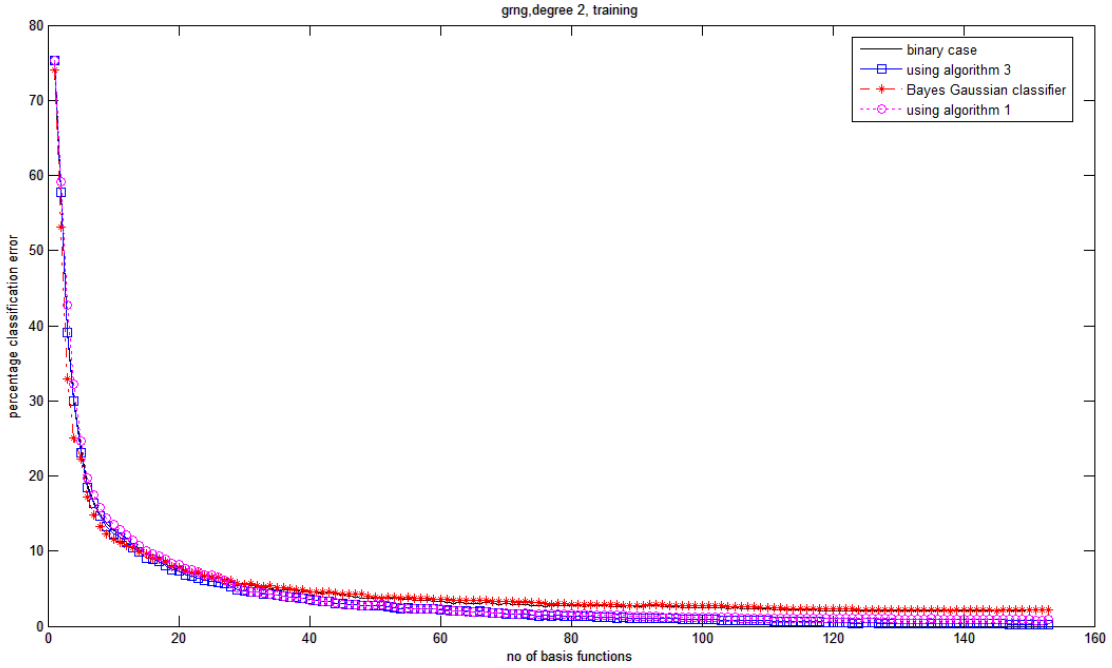

Figure 7.15 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 2.

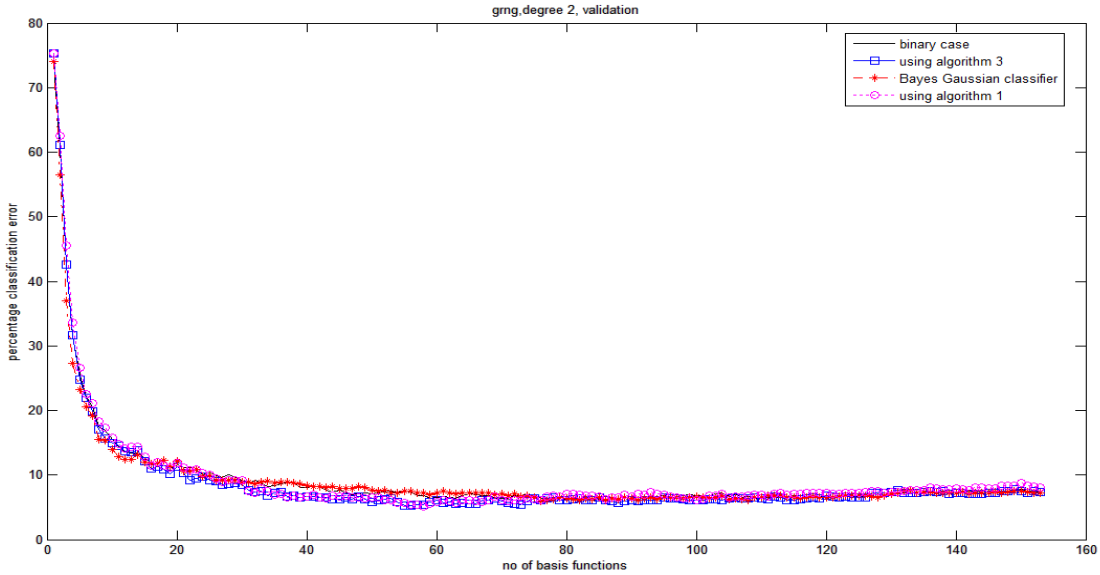

Figure 7.16 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 2.

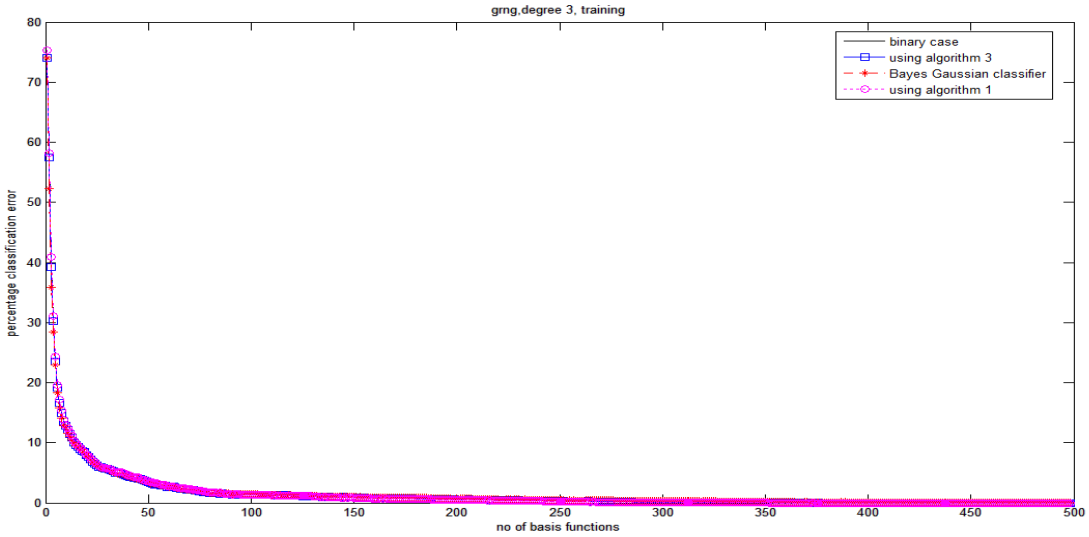

Figure 7.17 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 3.

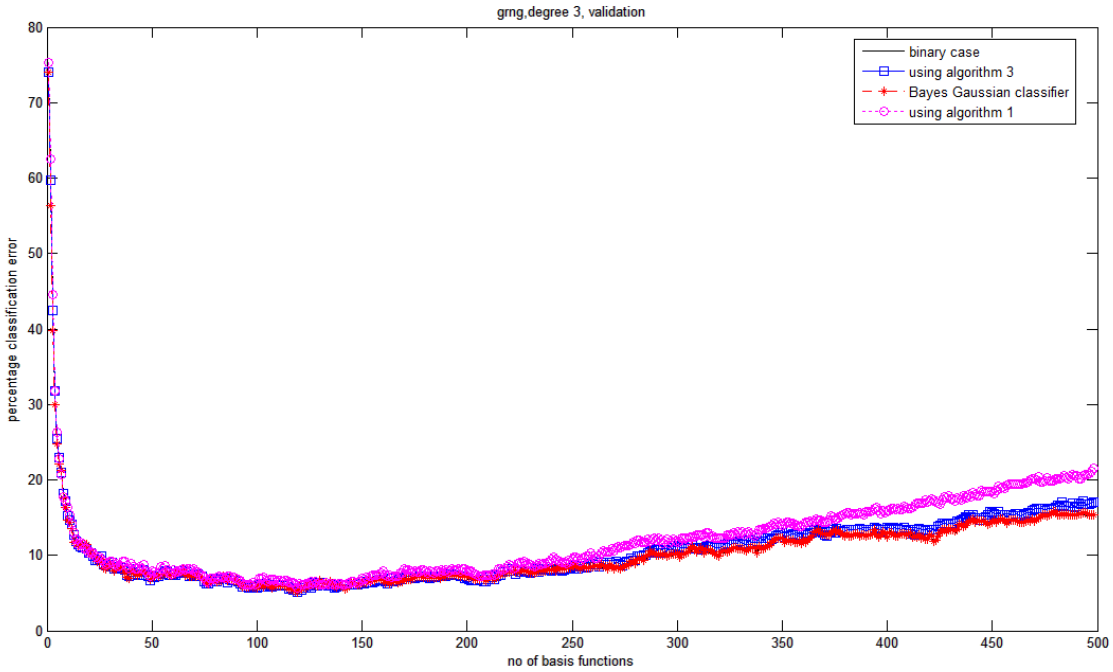

Figure 7.18 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 3.

7.4 Mushroom data file

This data set [22] includes descriptions of hypothetical samples corresponding to 23 species of grilled mushrooms in Agaricus and Lepiota family. Each sepecies is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The guide clearly states that there is no simple rule for determing the edibility of a mushroom.

Total number of patterns=8124

Training patterns=7312, Validation patterns=812

Inputs=22, Classes=2.

Table 7.4 Comparison of Percentage Classification Error for Mushroom data file

	Training			Validation		
	Degree 1	Degree 2	Degree 3	Degree 1	Degree 2	Degree 3
Algorithm 3	4.68346	0	0	4.696291	0	0
Algorithm 1	5.994664	0	0	6.035889	0	0
Bayes Gaussian Classifier	6.730899	0	0	6.711565	0	0
Binary case	6.512332	0	0	6.564525	0	0

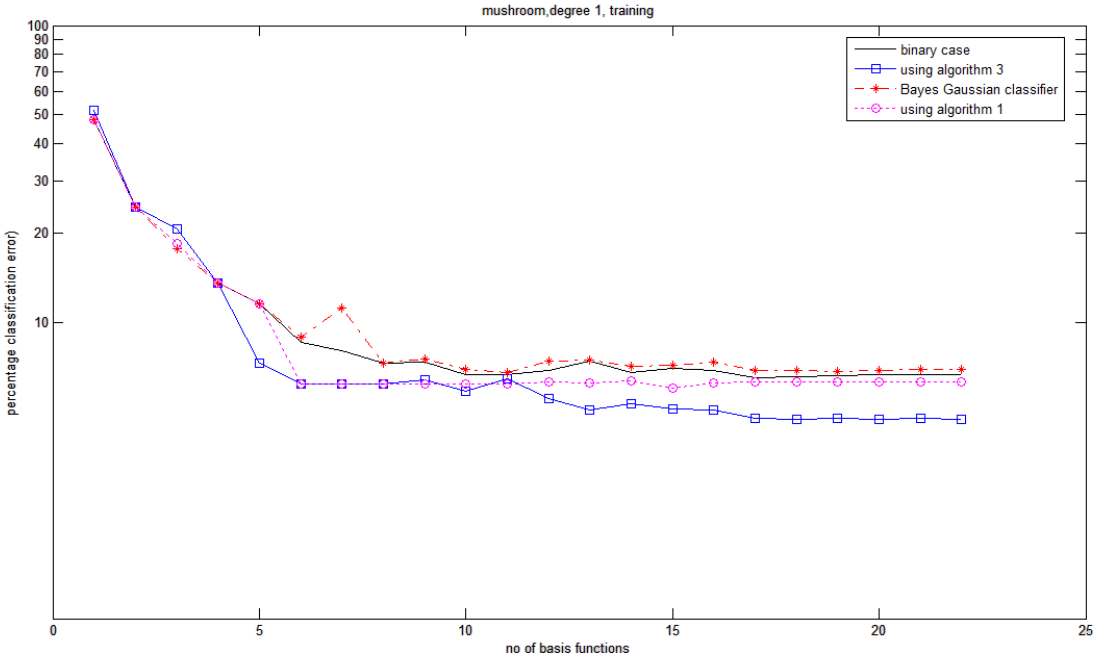

Figure 7.19 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 1.

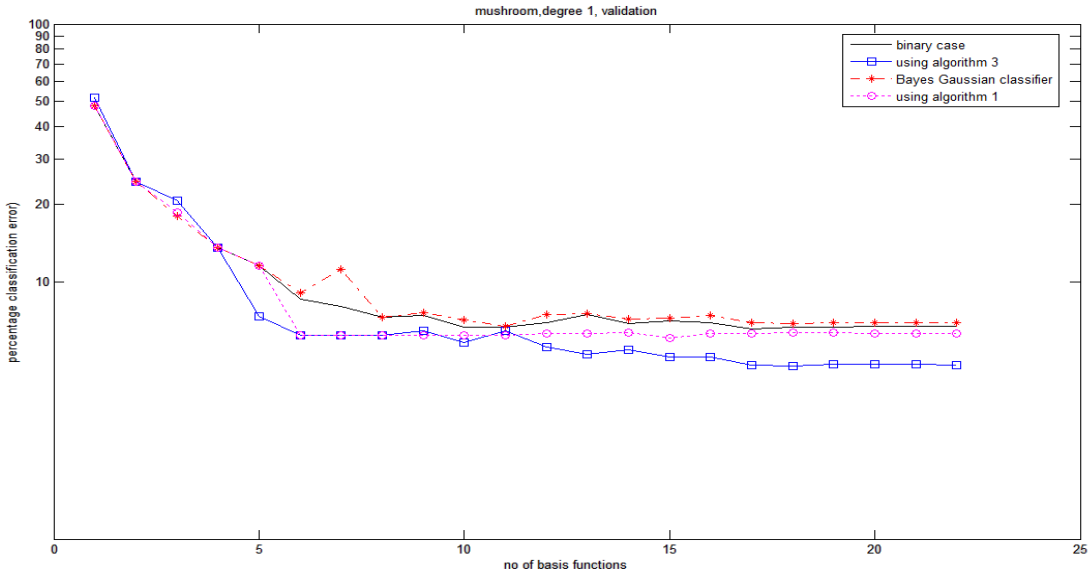

Figure 7.20 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 1.

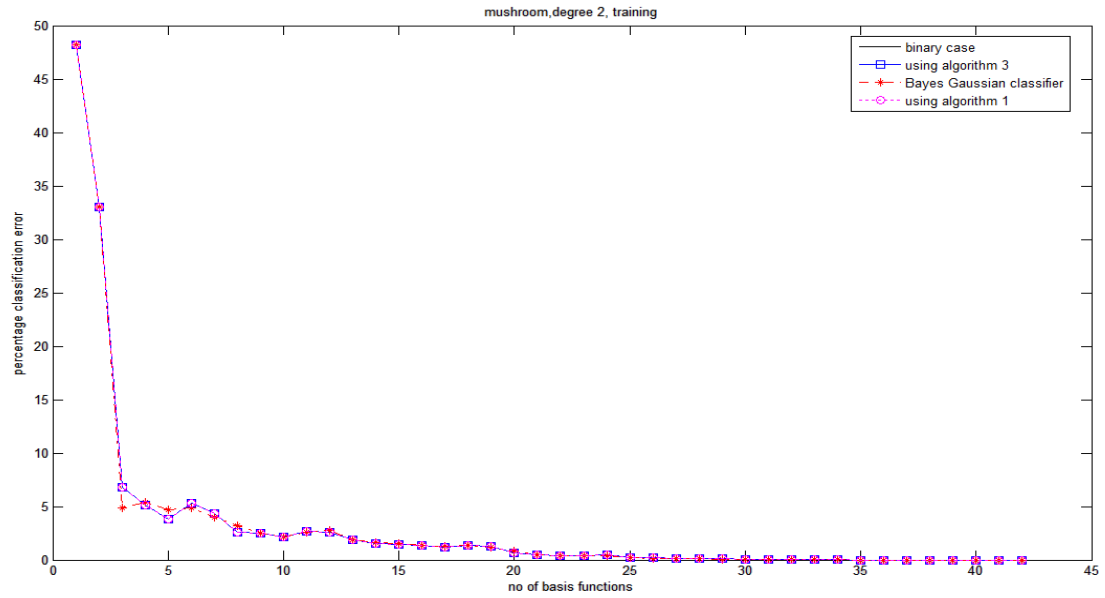

Figure 7.21 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 2.

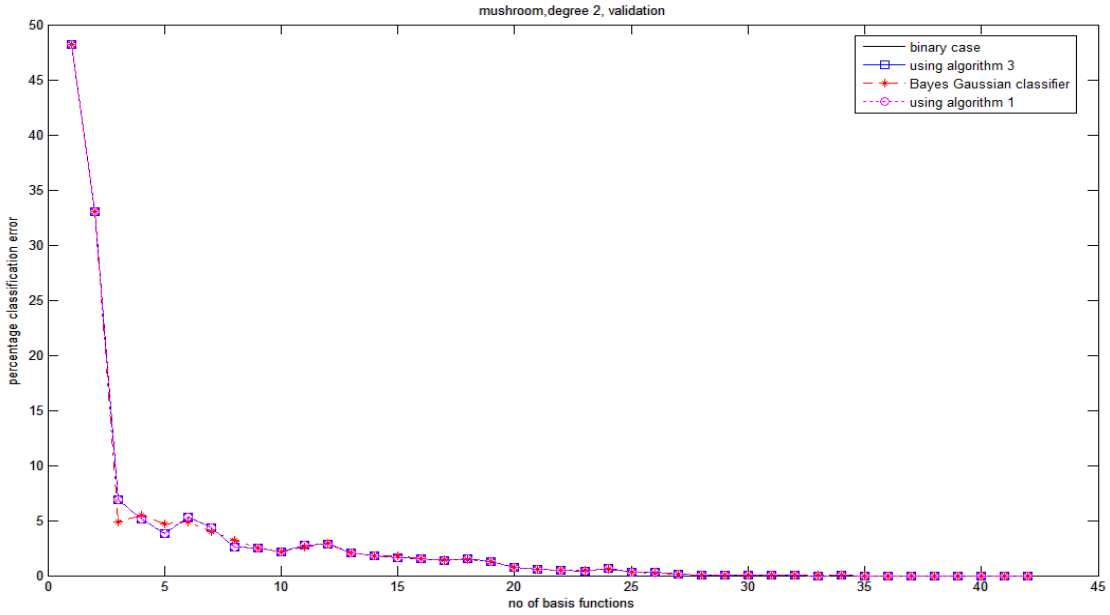

Figure 7.22 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 2.

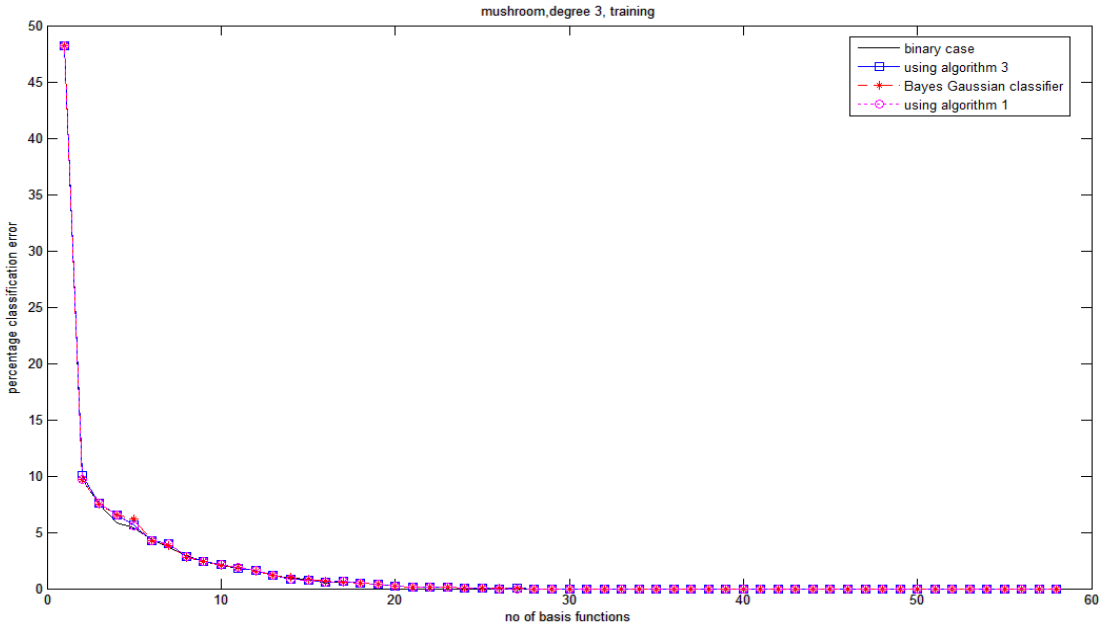

Figure 7.23 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 3.

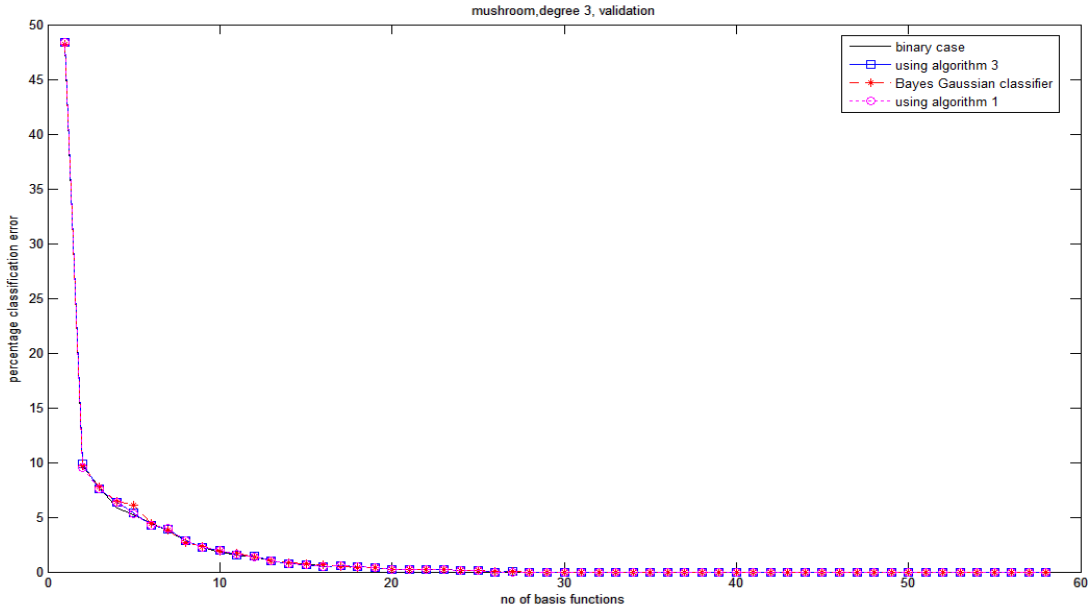

Figure 7.24 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 3.

7.5 Diabetes data file

Diabetes data set [22] contains the distribution for 70 sets of data recorded on diabetes patients (several weeks' to months' worth of glucose, insulin, and lifestyle data per patient + a description of the problem domain).

Total number of patterns=768

Training patterns= 692, Validation patterns= 76

Inputs=8, Classes=2.

Table 7.5 Comparison of Percentage Classification Error for Diabetes data file

	Training			Validation		
	Degree 1	Degree 2	Degree 3	Degree 1	Degree 2	Degree 3
Algorithm 3	21.66614	18.25711	11.8814	23.11688	23.37329	23.37329
Algorithm 1	21.96561	19.58368	14.09212	23.37329	22.60739	23.36663
Bayes Gaussian Classifier	22.43628	20.1256	15.73235	23.11688	22.34432	22.6024
Binary case	22.09396	19.68342	14.99075	23.24342	22.33933	22.85548

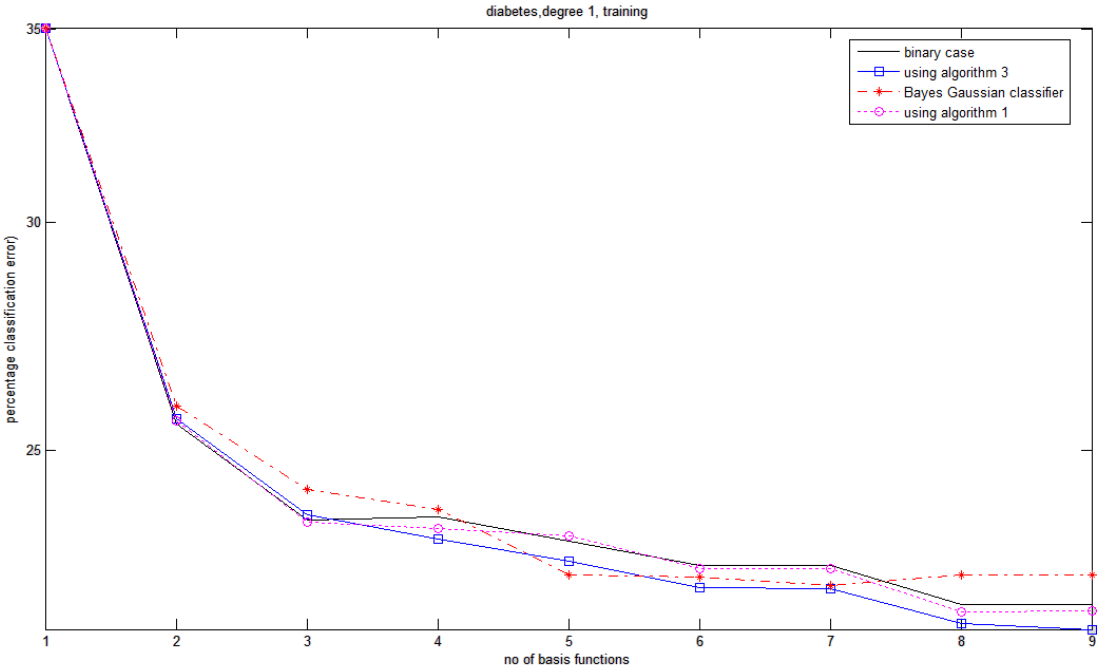

Figure 7.25 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 1.

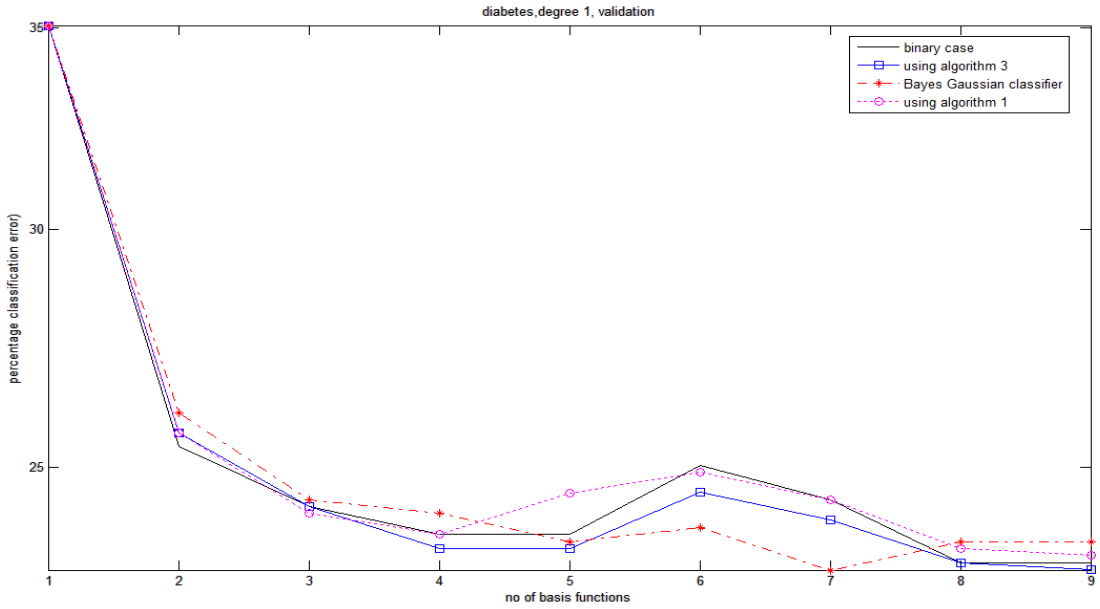

Figure 7.26 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 1.

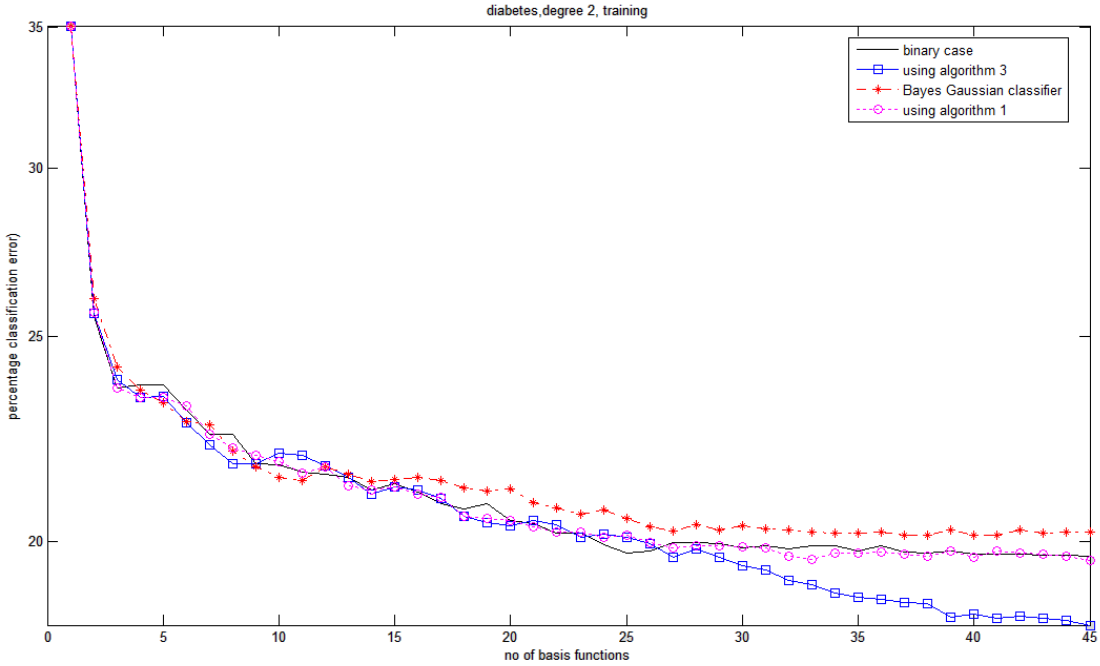

Figure 7.27 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 2.

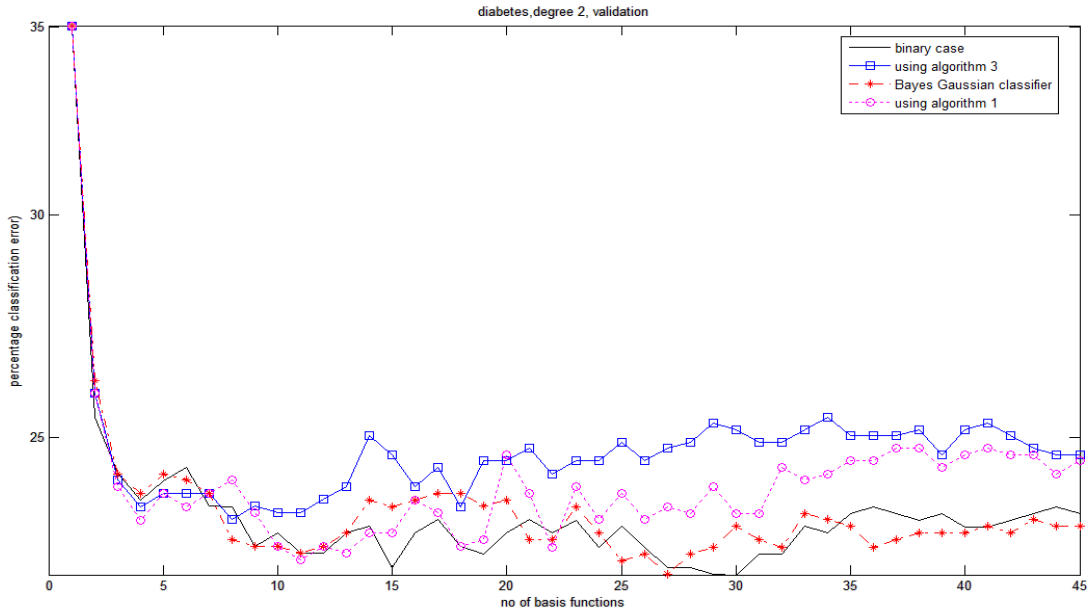

Figure 7.28 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 2.

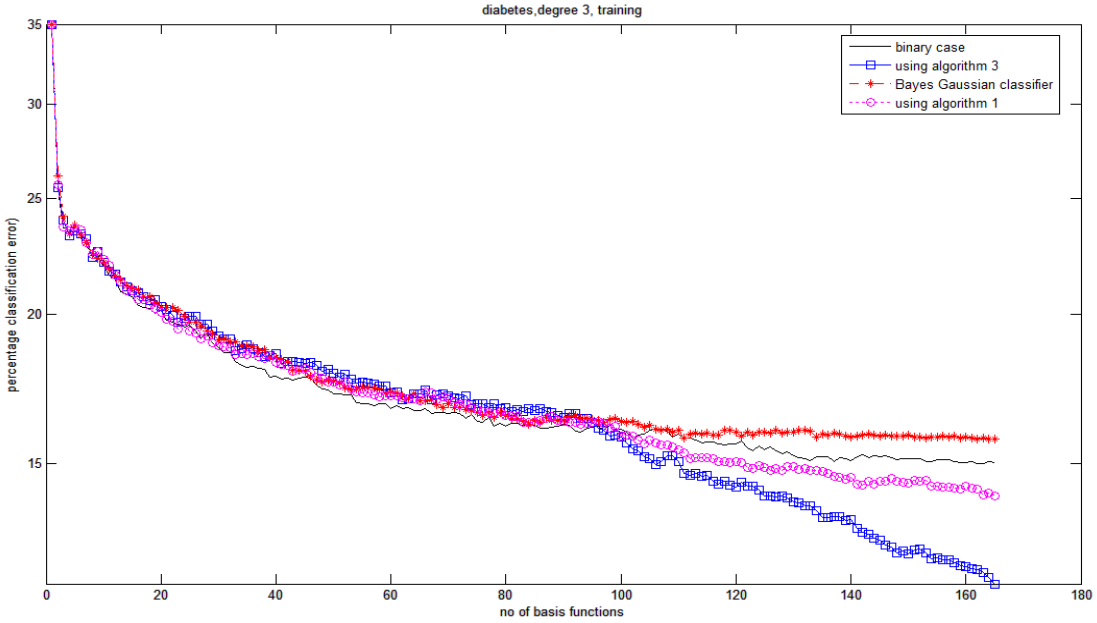

Figure 7.29 Training Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 3.

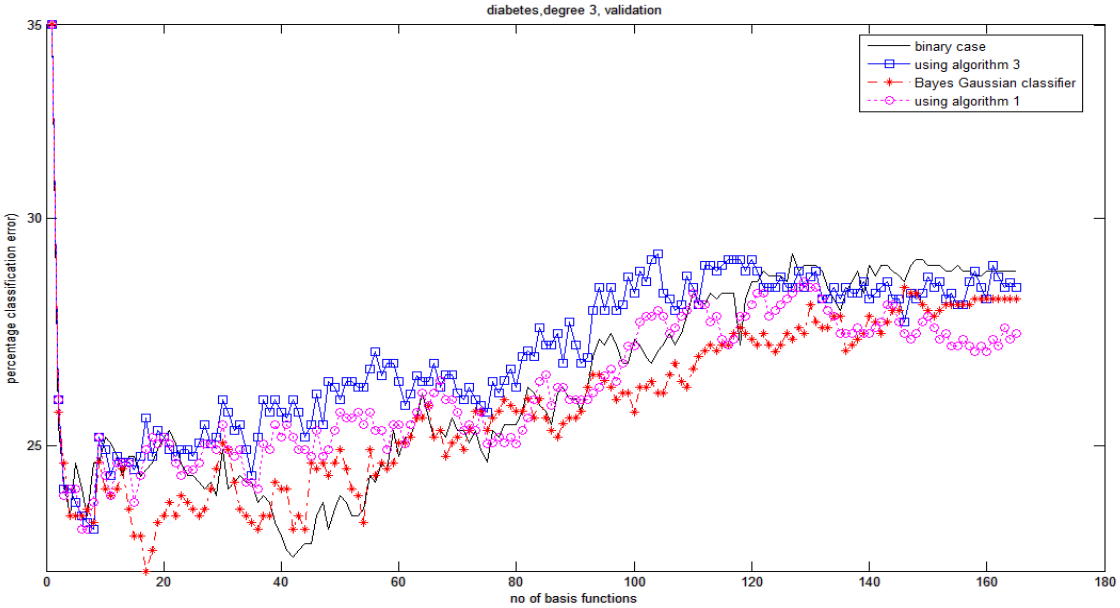

Figure 7.29 Validation Percentage Classification Error vs No. of Basis Functions Comparison of Algorithm 3, Algorithm 1, Bayes Gaussian Classifier and Binary Case for Degree 3.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In this document we prove that the Basic OR algorithm (Algorithm1) can handle correctly classified outliers but cannot handle incorrectly classified outliers. We also prove the convergence of this algorithm since the MSE in this algorithm is not proportional to probability of error; we develop a second algorithm in which the MSE is proportional to the probability of error and immune to outliers. Since the weights in this algorithm do not converge we fix this problem by developing Algorithm 3 in which correctly classified patterns also contribute to the MSE and the sum of the changes to the actual ouputs in a particular pattern are almost zero. We analyze weights changes for Algorithm 3.

Finally we compare the results of Algorithm 3, Algorithm 1 and a Bayes Gaussian classifier on both training and validation data on several different data sets. After comparison, algorithm 3 is found to work

In future we could do the following

(1) Algorithm 1 or Algorithm 3 can be tried for training multilayer perceptrons.

(2) Perhaps Output Reset can be used to design SVMs.

(3) Perhaps Output Reset can be used to design nonlinear networks using the L1 objective function.

APPENDIX A

GRAM SCHMIDT ALGORITHM

The Gram-Schmidt procedure maps a set of linearly dependent vectors to an orthonormal basis vector set in Euclidian or any inner product space. It is a well-known standard numerical method [25] and has been used to find optimum choice of Radial centers for RBF [10], fast computation of weights of RBF network [15], pruning of MLP [11] and feature selection in piecewise classifier [18]. What few authors did not realize is that using the Gram-Schmidt procedure allows ordering the basis function in order of their contribution to minimize the MSE. Using this, the network can be represented as a monotonically non-increasing function of addition of basis functions and achieve a faster rate of convergence. Also, orthonormalizing linearly dependent vectors results in a zero vector is an important step in pruning useless basis functions. These desirable properties along with the effective representation, system re-transformation and a fast and distributed iterative solution make it a better candidate over other learning algorithms.

Consider a vector $\mathbf{X}$ whose elements X_n are basis functions. Its corresponding orthonormal mapping is $\mathbf{X}'$ where $\mathbf{X}$ and $\mathbf{X}' \in \Re^N$. By the definition of orthonormality, vector $\mathbf{X}'$ is orthonormal only if it satisfies the following condition:

$$< X'(i)X'(j) >= \frac{1}{N_v}\sum_{p=1}^{N_v} X'_p(i)X'_p(j) = 0 \text{ if } i \neq j$$
$$= 1 \text{ if } i = j \qquad \text{(A.1)}$$

where $X'(i)$ is the i^{th} element of the vector $\mathbf{X}'$ and $< X'(i)X'(j)>$ denotes the inner product of $X'(i)$ and $X'(j)$. Here $X'_p(i)$ refers to the p^{th} example value of i^{th} orthonormal function $X'(i)$.

Also $\|X(i)\|$ is defined as

$$\| X(i) \| = \left[\frac{1}{N_v} \sum_{p=1}^{N_v} X_p(i)^2 \right]^{\frac{1}{2}} \tag{A.2}$$

Then by the standard Gram-Schmidt procedure, $X'(k)$ is calculated as:

$$X'(1) = \frac{X(1)}{\| X(1) \|} \tag{A.3}$$

$$X'(2) = \frac{X(2) - < X(2)X'(1) > X'(1)}{\| X(2) - < X(2)X'(1) > X'(1) \|} \tag{A.4}$$

……..

$$X'(k) = \frac{X(k) - \sum_{i=1}^{k-1} < X(k)X'(i) > X'(i)}{\| X(k) - \sum_{i=1}^{k-1} < X(k)X'(i) > X'(i) \|} \qquad \text{for } 1 \le k \le L \tag{A.5}$$

A.1 General approach towards Gram-Schmidt Procedure

An iterative solution to the Gram-Schmidt procedure is given here. **A** represents a lower triangular LxL orthonormal transformation matrix such that:

$$\mathbf{X}' = \mathbf{A} \cdot \mathbf{X} \tag{A.6}$$

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

$$X'(m) = \sum_{i=1}^{m} a(m,i)X(i) \quad \text{for } 1 \le m \le L \tag{A.7}$$

Let **R** be the auto-correlation matrix as defined earlier, where its elements r(i,j) is defined as

$$r(i,j) = \frac{1}{N_v} \sum_{p=1}^{N_v} X_p(i)X_p(j) \quad \text{for } 1 \le i,j \le L \tag{A.8}$$

Then from (A.6 – A.8),

$$a(1,1) = \frac{1}{\| X(1) \|} = \frac{1}{r(1,1)^{\frac{1}{2}}} \tag{A.9}$$

$$X'(1) = a(1,1) \cdot X(1) \tag{A.10}$$

$$X'(2) = a(2,1) \cdot X(1) + a(2,2) \cdot X(2) \tag{A.11}$$

Let $X'(2)$ be equal to

$$X'(2) = \frac{Z(2)}{\| Z(2) \|} \tag{A.12}$$

Then Numerator of $X'(1)$ is

$$Z(2) = X(2) - b(1) \cdot X'(1) = X(2) - b(1) \cdot a(1,1) \cdot X(1) \tag{A.13}$$

where

$$b(1) = < X'(1) \cdot X(2) > = a(1,1) \cdot r(1,1) \tag{A.14}$$

Writing Z(2) as

$$Z(2) = \sum_{k=1}^{2} c(k)X(k) = c(1)X(1) + c(2)X(2) \tag{A.15}$$

Here,

$$c(1) = -b(1)a(1,1) \tag{A.16}$$

$$c(2)=1 \tag{A.17}$$

Also,

$$\| Z(2) \| = \| < X(2) - b(1)X'(1), X(2) - b(1)X'(1) > \| \tag{A.18}$$

$$\therefore \| Z(2) \| = \left[r(2,2) - b(1)^2 \right]^{\frac{1}{2}} \tag{A.19}$$

Equating (A.12) and (A.15)

$$X'(2) = [a(2,1)X(1) + a(2,2)X(2)] = \frac{c(1)X(1) + c(2)X(2)}{[r(2,2) - b(1)^2]^{\frac{1}{2}}} \tag{A.20}$$

$$a(2,1) = \frac{-b(1)a(1,1)}{[r(2,2) - b(1)^2]^{\frac{1}{2}}} \tag{A.21}$$

$$a(2,2) = \frac{1}{[r(2,2) - b(1)^2]^{\frac{1}{2}}} \tag{A.22}$$

An iterative approach for finding the a(i,j) coefficients can be extended as follows:

For $1 \leq m \leq L$, perform the following operations

$$b(i) = \sum_{q=1}^{i} a(i,q) \cdot r(q,m) \quad \text{for} \quad 1 \leq i \leq m\text{-}1 \tag{A.23}$$

$$c(m) = 1 \tag{A.24}$$

$$c(k) = -\sum_{i=k}^{m-1} b(i) \cdot a(i,k) \quad \text{for} \quad 1 \leq k \leq m\text{-}1 \tag{A.25}$$

$$a(m,k) = \frac{c(k)}{[r(m,m) - \sum_{i=1}^{m-1} b^2(i)]^{\frac{1}{2}}} \quad \text{for} \quad 1 \le k \le m \tag{A.26}$$

A.2 Solving for the Orthonormal System Weights

When the basis functions **X** are transformed to **X**′, the system is mapped into new weights w′(k,i) for the k^{th} estimated output $y_p(k)$ and i^{th} orthonormal basis function X′(i) for given training dataset with N_v patterns.

$$y_p(k) = \sum_{i=1}^{L} w'(k,i) \cdot X'_p(i) \tag{A.27}$$

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

$$E_k = \frac{1}{N_v} \sum_{p=1}^{N_v} [y_p(k) - \sum_{i=1}^{L} w'(k,i) \cdot X'_p(i)]^2 \quad \text{for } 1 \le k \le M. \tag{A.28}$$

Taking partial derivative w.r.t. the variable w′(k,m), L equations in L unknowns are obtained, thus there would be a unique solution with only a global minimum. Using (A.7) the orthonormal weights are given by

$$w'(k,m) = \sum_{i=1}^{m} a(m,i) \cdot c(k,i) \quad \text{for } 1 \le k \le M,\ 1 \le m \le L \tag{A.29}$$

where c(k,i) is an element of the cross-correlation matrix (**C**).

A.3 Re-mapping Weights from Orthonormal System

To understand the mapping relationship in terms of the original system and to avoid additional computation of orthonormalizing the basis functions for validation and real-time processing, it is important to represent the system in terms of the original weights. An efficient mapping orthonormal weights to the original system weights is achieved by equating Eq. (2) and (A.28) and then substituting for $\mathbf{X}'$ from (A.7)

$$w(k,i) = \sum_{j=1}^{L} w'(k, j) \cdot a(i, j) \tag{A.31}$$

APPENDIX B

BAYES GAUSSIAN CLASSIFIER

The Bayes Gaussian Classifier [20] is a Bayes classifier where the conditional pdf f(**X**|i) is assumed to be Gaussian. Most of the data available in the real world is approximately Gaussian because of the "Central Limit Theorem" [19], so this classifier is often applicable.

B.1 Derivation of Bayes discriminant

The conditional probability density is given as,

$$f(\mathbf{X} \mid i) = \frac{1}{(2\pi)^{\frac{N}{2}} \mid \mathbf{C}_i \mid^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{X}-\mathbf{m}_i)^T \mathbf{A}_i (\mathbf{X}-\mathbf{m}_i)} \qquad \text{for } 1 \le i \le N_c \tag{B.1}$$

where $\mathbf{A}_i$ is the inverse covariance matrix and $\mathbf{C}_i$ is the covariance matrix calculated as,

$$C_i(n,m) = \frac{1}{N_v(i)} \sum_{p:i_c(p)=i}^{N_v(i)} [X_p(n) - m_i(n)][X_p(m) - m_i(m)] \quad \text{for } 1 \le n,m \le N \tag{B.2}$$

Here $N_v(i)$ is the number of patterns belonging to the i^{th} class and $\mathbf{m}_i$ is the mean input vector belonging to the i^{th} class and is calculated as,

$$m_i(n) = \frac{1}{N_v(i)} \sum_{p:i_c(p)=i}^{N_v(i)} X_p(n) \tag{B.3}$$

The Bayes discriminant [20] is calculated as,

$$d(i) = -2 \ln \left[\frac{P_i \cdot \exp[-\frac{1}{2}(\mathbf{X}_p - \mathbf{m}_i)^T \mathbf{A}_i (\mathbf{X}_p - \mathbf{m}_i)]}{(2\pi)^{\frac{N}{2}} \mid \mathbf{C}_i \mid^{\frac{1}{2}}} \right] \quad \text{for } 1 \le i \le N_c \tag{B.4}$$

where P_i is the probability of occurrence for the i^{th} class and $N_c = M$ = number of outputs = number of classes.

Eq. (B.4) can be rewritten as

$$d(i) = (\mathbf{X}_p - \mathbf{m}_i)^T \mathbf{A}_i (\mathbf{X}_p - \mathbf{m}_i) + B_i \quad \text{for } 1 \le i \le N_c \tag{B.5}$$

Where

$$B_i = N \cdot \ln(2\pi)^{\frac{N}{2}} + \ln|\mathbf{C}_i| - 2 \cdot \ln(P_i)$$

If we assume the same covariance matrix for each class then

$$d(i) = \mathbf{X}_p{}^T \cdot \mathbf{A} \cdot \mathbf{X}_p + \mathbf{m}_i{}^T \cdot \mathbf{A} \cdot \mathbf{m}_i - \mathbf{m}_i{}^T \cdot \mathbf{A} \cdot \mathbf{X}_p - \mathbf{X}_p{}^T \cdot \mathbf{A} \cdot \mathbf{m}_i - 2 \cdot \ln(P_i) + \ln|\mathbf{C}| + N \cdot \ln(2\pi)^{\frac{N}{2}} \tag{B.6}$$

$$d(i) = -2 \cdot \mathbf{m}_i{}^T \cdot \mathbf{A} \cdot \mathbf{X}_p + [\mathbf{m}_i{}^T \cdot \mathbf{A} \cdot \mathbf{m}_i - 2 \cdot \ln(P_i)] \quad \text{for } 1 \le i \le N_c \tag{B.7}$$

we get

$$d(i) = \mathbf{w_i} \cdot \mathbf{X}_p + k_i \quad \text{for } 1 \le i \le N_c \tag{B.8}$$

Where

$\mathbf{w}_i = -2 \cdot \mathbf{m}_i{}^T \cdot \mathbf{A}$ and

$k_i = \mathbf{m}_i{}^T \cdot \mathbf{A} \cdot \mathbf{m}_i - 2 \cdot \ln(P_i)$

We find the value of i such that d(i) is minimum. The resulting value of i is our estimate of i_c.

In Eq. (B.1 - B.8) instead of $\mathbf{X}_p$ we could use $\mathbf{X}'_p$ since $\mathbf{X}'_p$ is orthonormal transformation of $\mathbf{X}_p$. Since we are using $\mathbf{X}'_p$ all the basis functions are orthonormal to each other as in Eq. (A.1) so the covariance matrix will be a diagonal matrix for the

entire data file, so we will be using the same covariance matrix (diagonal matrix) for all the classes which is an assumption. So Eq. (B.8) is now rewritten as

$$d(i) = \mathbf{w}_{\mathbf{i}}' \cdot \mathbf{X}_p + k_i' \quad \text{for } 1 \le i \le N_c \tag{B.9}$$

Where

$\mathbf{w}_i' = -2 \cdot \mathbf{m}_{\mathbf{i}}$ and

$$k_i' = \mathbf{m}_i^{\,T} \cdot \mathbf{m}_i - 2 \cdot \ln(P_i) = \sum_{n=1}^{L} [m_i(n)]^2 - 2 \cdot \ln(P_i)$$

REFERENCES

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

[2] K. Hornik, M. Stinchcombe, and H. White, “Multilayer Feedforward Networks Are Universal Approximators.” Neural Networks, Vol. 2, No. 5, 1989, pp. 359-366.

[3] K. Hornik, M. Stinchcombe, and H. White, “Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforward Networks,” Neural Networks, vol. 3, 1990, pp. 551-560.

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

[5] Dennis W. Ruck, Steven K. Rogers, Matthew Kabrisky, Mark E. Oxley and Bruce W. Suter, “The Multilayer Perceptron as an approximation to a Bayes optimal discriminant function,” *IEEE Trans Neural Networks*, TNN-1(4):296-298, 1990.

[6] Y. H. Pao and Y. Takefuji, “Functional-Link Net Computing: Theory, System Architecture, and Functionalities,” IEEE Computer, Vol. 25, No. 5, 1992, pp. 76 - 79.

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

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

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

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

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

[12] Saurabh Sureka, Michael T. Manry, “A Functional Link Network With Ordered Basis Functions”. IJCNN 2007: pp 1708-1713.

[13] R.G. Gore, J.Li, M.T. Manry, L. M. Liu, C. Yu and J.Wei, ”Iterative Design Of Neural Network Classifiers Through Regression”,International Journal on Artificial Intelligence Tools, Vol. 14, Nos. 1 & 2 (2005) ,281-301

[14] L.M. Liu, M.T. Manry, F. Amar, M.S. Dawson, and A.K. Fung, “Iterative Improvement of Image Classifiers Using Relaxation,” 28th ASILOMAR Conference on Signals, Systems and Computers, Vol. 2, pp.902-906, 1995.

[15] H.C.Yau and M. T. Manry, “Iterative Improvement of a Nearest Neighbor Classifier,” Neural Networks, Vol.4, Number 4, pp.517-524, 1991.

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

[17] R. R. Bailey, E. J. Pettit, R. T. Borochoff, M. T. Manry, and X. Jiang, "Automatic Recognition of USGS Land Use/Cover Categories Using Statistical and Neural Network Classifiers," Proceedings of SPIE OE/Aerospace and Remote Sensing, April 12-16, 1993, Orlando Florida.

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

[19] Athanasios Papoulis, "Probability, Random Variables, and Stochastic Processes", second edition. New York: McGraw- Hill.

[20] Jimy Shah, M. T. Manry, "Sequences of Bayes Gaussian Classifiers", Univeristy of Texas, Arlington, December 2007.

[21] Pramod Lakshmi Narasimha, M. T. Manry, "Sequences of Near-Optimal Feed Forward Neural Networks", Univeristy of Texas, Arlington, August 2007.

[22] D. N .A. Asuncion, "UCI machine learning repository," 2007 [Online]. Available:http://www.ics.uci.edu/~mlearn/MLRepository.html

[23] Burgess, C. (1998). From simple associations to the building blocks of language: Modeling meaning in memory with the HAL model. *Behavior Research Methods, Instruments, & Computers, 30,* 188 - 198.

[24] Bishop CM, Tipping ME. Bayesian regression and classification. Suykens J Horvath G Basu S Micchelli C Vandewalle J eds. *Advances in Learning Theory: Methods, Models and Applications(NATO Science Series. Series III, Computer and Systems Sciences Vol. 190).* 2003;267–285. IOS Press Amsterdam.

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

BIOGRAPHICAL INFORMATION

Madhu Gannapal received his Bachelor of Engineering from Sri Jayachamarajendra College of Engineering (SJCE), Mysore India in 2007. He obtained his Master's degree from the University of Texas at Arlington, USA in May 2010 under the Department of Electrical Engineering. His current research interests include image processing, pattern recognition, software development and speech recognition.