Name : Achyut Kulkarni

Solutions :

Bias Variance Trade-off :

1)

a)According to Affine Transformation property of Multivariate Gaussian distribution,

$$y \tilde{} N(c + B\mu,\ B\Sigma B^T)$$

Substituting the corresponding values of $\hat{\beta}$ equation,

Mean $$c + B\mu = (X^T X + \lambda I)^{-1} X^T X\beta^* + (X^T X + \lambda I)^{-1} X^T \mu$$

Variance $$B\Sigma B^T = (X^T X + \lambda I)^{-1} X^T \Sigma ((X^T X + \lambda I)^{-1} X^T)^T$$

$$= (X^T X + \lambda I)^{-1} X^T \Sigma X (X^T X + \lambda I)$$

Thus $\hat{\beta}_\lambda$ distribution is,

$$\hat{\beta}_\lambda \tilde{} N((X^T X + \lambda I)^{-1} X^T X\beta^* + (X^T X + \lambda I)^{-1} X^T \mu, (X^T X + \lambda I)^{-1} X^T \Sigma X (X^T X + \lambda I))$$

1b) Calculating Bias,

$$Bias = E[x^T \hat{\beta}_\lambda] - x^T \beta^\star$$

$$Bias = x^T E[\hat{\beta}_\lambda] - x^T \beta^\star$$ 1) Closed for solution for $\hat{\beta}$ *is*,

$$\hat{\beta}_\lambda = (X^T X + \lambda I)^{-1} X^T y$$

Substituting given y, we will get the following closed form solution,

$$\hat{\beta}_\lambda = (X^T X + \lambda I)^{-1} X^T (X\beta^* + \epsilon)$$

To find $\hat{\beta}_\lambda$ *distribution*,

$$\hat{\beta}_\lambda = (X^T X + \lambda I)^{-1} X^T X\beta^* + (X^T X + \lambda I)^{-1} X^T \epsilon$$

We know that ϵ comes from some Multivariate Gaussian Distribution $N(\mu, \Sigma)$

We can observe that the above equation can be mapped to the following form

$$y = c + Bx$$

where $y = \hat{\beta}_\lambda$, $c = (X^T X + \lambda I)^{-1} X^T X \beta^*$, $B = (X^T X + \lambda I)^{-1} X^T$, $x = \epsilon$ and x~ $N(\mu, \Sigma)$

Substituting the values of $\hat{\beta}$,

$$Bias = x^T E[(X^T X + \lambda I)^{-1} X^T y] - x^T \beta^\star$$

$$Bias = x^T((X^T X + \lambda I)^{-1} X^T E[y]) - x^T \beta^\star$$

$$Bias = x^T((X^T X + \lambda I)^{-1} X^T (X\beta^\star)) - x^T \beta^\star$$

$$Bias = x^T((X^T X + \lambda I)^{-1} X^T (X\beta^\star) - \beta^\star)$$

$$Bias = x^T((X^T X + \lambda I)^{-1} X^T X - I)\, \beta^\star$$

1)

c) Variance

We see that $x^T (\hat{\beta}_\lambda - E [\hat{\beta}_\lambda])$ is a gaussian distribution with variance $x^T . (X^T X + \lambda I)^{-1} X^T \ X(X^T X + \lambda I) . x$

Square of a gaussian random variable is a χ^2 variable.

Mean of a χ^2 distribution is $\left\| X . (X^T X + \lambda I)^{-1} . x \right\|^2$

Thus variance $E[\ (x^T (\hat{\beta}_\lambda - E [\hat{\beta}_\lambda]))^2] = \left\| X . (X^T X + \lambda I)^{-1} . x \right\|^2$

1d)

The bias and variance trade of can be written as

$$Error = Bias^2 + Variance + Noise$$

$$Error = (x^T((X^T X + \lambda I)^{-1} X^T X - I)\beta^\star)^2 + variance + const$$

We can observe that as λ increases the bias increases and the variance decreases.

When λ is small, bias is dominating

When λ is large, variance is dominating

2) Kernel Construction

Given: $k_1(\cdot,\cdot)$ and $k_2(\cdot,\cdot)$ are kernel functions

Mercer's theorem states a bivariate function k(·, ·) is a positive definite kernel function
Iff the matrix K { $K_{ij} = k(x_i, x_j)$ } is positive semi definite.
K is positive semidefinite if all its eigenvalues are Non-Negative.

K_n is of the Matrix of the form $[\,[\,k_n(x,x),\ k_n(x,x')\,],\,[\,k_n(x',x),\ k_n(x',x')\,]\,]$ where k is some kernel function

(a) K_1 and K_2 positive semi definite as k_1 and k_2 are valid kernel functions:
From the theorems of Positive definite Matrices:
If A, B are positive semidefinite matrices then A+B is also a positive definite matrix , hence;

Let $K' = K_1 + K_2$

For any $a_1 > 0$ a_1K_1 will also be positive semi definite

$K_3 = a_1K_1 + a_2K_2$ is positive semi definite given $a_1 > 0$ and $a_2 > 0$

Thus,

$k_3(x,y) = a_1k_1(x,y) + a_2k_2(x,y)$ will be a valid kernel function

(b) Here w.k.t

f(·) is a real valued function.

$$k_4(x,x') = f(x)\,.\,f(x')$$

We can now calculate the eigenvalues for the function by the definition :

$$y = det(K_4 - I\lambda) = 0$$

Since we know ,

K_n is of the Matrix of the form $[\,[\,k_n(x,x),\ k_n(x,x')\,],\,[\,k_n(x',x),\ k_n(x',x')\,]\,]$ thus,

$$y = (f(x)^2 - \lambda)\,.\,(f(x')^2 - \lambda) - (f(x)\,.\,f(x'))\,.\,(f(x)\,.\,f(x')) = 0$$

$$y = \lambda^2 - \lambda(f(x)^2 + f(x')^2) + D = 0$$

Thus calculating the extremum to find λ :

$$dy/d\lambda = 2\lambda - (f(x)^2 + f(x')^2) = 0$$

$$\lambda^* = (f(x)^2 + f(x')^2)/2$$

is an extremum and thus we need to identify if its minima, hence;
$d^2y / d\lambda^2 = 2$ which is > 0 thus function y is concave up and λ^* is the minimum.

The minimum possible value of (λ) $\lambda^* \geq 0$, thus $\lambda \geq 0$.
Thus K_4 is positive semi definite hence k_4 is a valid kernel function.

(c) $k_5(x,x') = k_1(x,x') \, . \, k_2(x,x')$
From the theorems of Positive definite matrices and concept of Inner product of positive definite matrices we know that ;
The inner product of two positive semidefinite matrices is also a positive semidefinite matrix

$$K_5 = K_1 \circ K_2$$

Hence , using this concept ; K_5 is positive semidefinite and
Thus k_5 is a valid kernel function.

3) Kernel Regression

Given:

The cost function of Linear Ridge regression is :

$$min_w \sum_n (y_i - w^T x_i)^2 + \lambda ||w||_2^2 \qquad \text{where } \lambda >= 0 \qquad \text{---------- (1)}$$

3)a)

Solution:

Differentiating (1) and equating to 0 we get,

$$\frac{\partial(1)}{\partial w} = 2\sum_n (y_i - w^T x_i)(-x_i) + 2\lambda w = 0$$

$$\sum_n (y_i - w^T x_i)(-x_i) + \lambda w = 0$$

$$\sum_n (y_i x_i) = \lambda w + \sum_n x_i x_i^T w$$

$$\sum_n (y_i x_i) = \lambda (Iw) + \sum_n x_i x_i^T w$$

As we can write Any matrix as the product of itself and the Identity Matrix

I.,e W = I*W

But we know that $\sum_n (y_i) = Y$, $\sum_n (x_i x_j) = X$ *and thus the equation finally reduces to,*

$$X^T Y = (X^T X + \lambda I) w$$

$$w = (X^T X + \lambda I)^{-1} X^T Y$$

b)

Given:

Φ is the nonlinear feature mapping which is a kernel function applied on each x_i

The cost function is thus:

$$min_w \sum_n (y_i - w^T \Phi(x_i))^2 + \lambda ||w||_2^2 \quad \text{--- ------- (2)}$$

Solution:

Differentiating (2) and equating to 0 ,

$$\frac{\partial(2)}{\partial w} = 2\sum_n (y_i - w^T \Phi(x_i))(-\Phi(x_i)) + 2\lambda w = 0$$

$$\sum_n (y_i - w^T \Phi(x_i))(-\Phi(x_i)) + \lambda w = 0$$

$$\sum_n (y_i \Phi(x_i)) = \lambda w + \sum_n \Phi(x_i)\Phi^T(x_i)w$$

$$\sum_n (y_i \Phi(x_i)) = \lambda(Iw) + \sum_n \Phi(x_i)\Phi^T(x_i)w$$

As we can write Any matrix as the product of itself and the Identity Matrix

I.,e W = I*W

But we know that $\sum_n (y_i) = Y$, $\sum_n \Phi(x_i)\Phi(x_i) = \Phi$ *and thus the equation finally reduces to*,

$$\Phi^T Y = (\Phi\Phi^T + \lambda I_N)w$$

$$w = (\Phi\Phi^T + \lambda I_N)^{-1}\Phi^T Y$$

By matrix inversion lemma we get,

$$w = \Phi^T(\Phi\Phi^T + \lambda I_N)^{-1} Y$$

c)

Given:

K is kernel matrix which is defined as $K_{ij} = \Phi_i^T \Phi_j$, $f' = w^{*T}\Phi(x)$

Solution

We know that,

From our previous definition of w , w => w^* ;

$$w^* = \Phi^T(\Phi\Phi^T + \lambda I_N)^{-1} Y$$

Hence,

$$f' = w^{*T}\Phi(x)$$

And we know that ;

$$w^{*T} = [\Phi^T(\Phi\Phi^T + \lambda I_N)^{-1} Y]^T$$

$$w^{*T} = Y^{T}(\Phi^{T}\Phi + \lambda I_N)^{-1}\Phi^{T}$$

$$w^{*T}\Phi(x) = Y^{T}(\Phi^{T}\Phi + \lambda I_N)^{-1}\Phi^{T}\Phi(x)$$

$$w^{*T}\Phi(x) = Y^{T}(K + \lambda I_N)^{-1}k(x)$$

As we know that $K = \Phi^{T}\Phi$ and $k(x) = \Phi^{T}\Phi(x)$ ..
Hence ;

$$f' = Y^{T}(K + \lambda I_N)^{-1}k(x)$$

3d)

We know that Matrix Multiplication of two matrices of order ;
A = (p*q) and B = (q*n) then A*B = (p*n) and is of order O(p*q*n)
Or it is generally of order O(n^3)

Now.. w.k.t ;

Computational complexity of linear regression is O($ND^2 + D^3$)
Linear regression :

Parameters : $w = (X^{T}X + \lambda I)^{-1}X^{T}Y$

Now,

$$X^{T}Y \Rightarrow O(ND)$$
$$(X^{T}X + \lambda I)^{-1} \Rightarrow O(D^3)$$
$$X^{T}Y \Rightarrow O(D^2)$$
$$X^{T}X \Rightarrow O(ND^2)$$

Computational complexity of kernel regression is O($N^3 + TN^2$)
Kernel Ridge Regression :

Parameters : $w = \Phi^{T}(\Phi\Phi^{T} + \lambda I_N)^{-1}Y$ å

$$\Phi\Phi^{T} \Rightarrow O(ND^2)$$
$$(\Phi^{T}\Phi + \lambda I_N)^{-1} \Rightarrow O(N^3)$$
$$\Phi^{T} \Rightarrow O(TN^2)$$

4) Support Vector Machines :

The positive examples are (1, 1) and (−1, −1). The negative examples are (1, −1) and (−1, 1).

Solutions

a) No

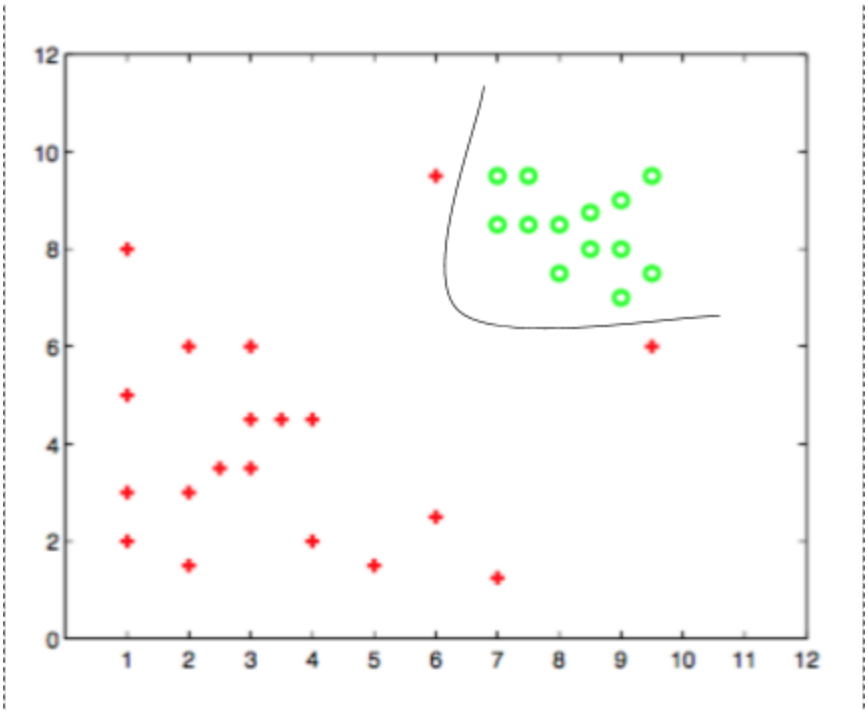
b) Φ(x) = [1, x_1 , x_2 , x_1 x_2] where x_1 x_2 are first and second coordinates of x :
The coordinates are (1,1,1,1) , (1,-1, -1,1) and (1,1,-1,-1) (1,-1,1,-1)
Hence We can easily see that final coordinate of each point helps in decision boundary
Therefore : Parameter :
W = [0,0,0,1]

c) Adding the point [0.5,-0.5] in the positive class makes it linearly Non-separable

d) The Kernel K(x,x) corresponds to a Polynomial of Degree 2 or a quadratic Kernel as it has the product of x_1 x_2 in the kernel function and hence it is of degree two.

5) SVMs and the slack penalty C :

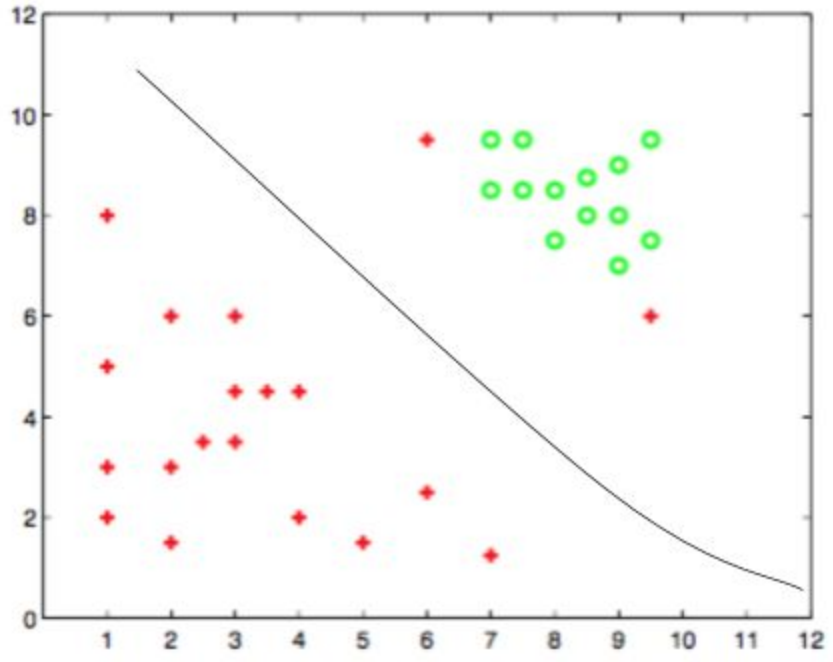
a) As $C -> \infty$, The Quadratic kernel we chose tries to fit the data more accurately and the decision boundary for SVM - a quadratic curve divides the two classes with more precision. This increases the Variance and thus ends up ***overfitting*** the curve.



b)

As $C -> 0$, The Quadratic kernel we chose tries to fit the bulk of the data with a smoother curve and generalises a curve which is close to a linear line. However it does not classify all the points perfectly during training and hence suffers from ***Underfitting*** as the value of C keeps decreasing. This also means the curve generalises and is Biased and hence the Bias increases as C keeps reducing

`

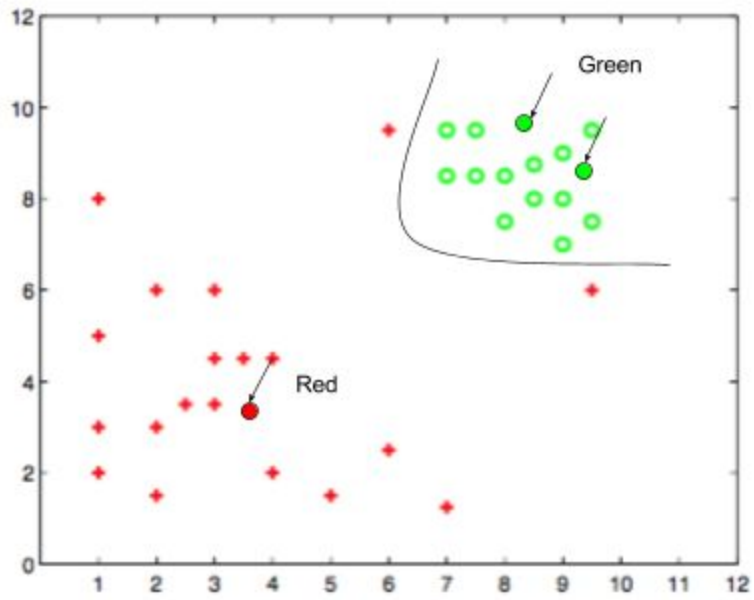


c)

Here, as from the given it is clear that the training data is produced by the sensors which are error prone and hence can produce some points which could be classified wrongly. So, in Such cases it is better to choose the Second Case (High Bias case) as , it provides better classification generalisation as it doesn't classify outliers which could be error prone unlike the First Case (high variance Case) which tries to even classify error prone outliers to fit the curve .

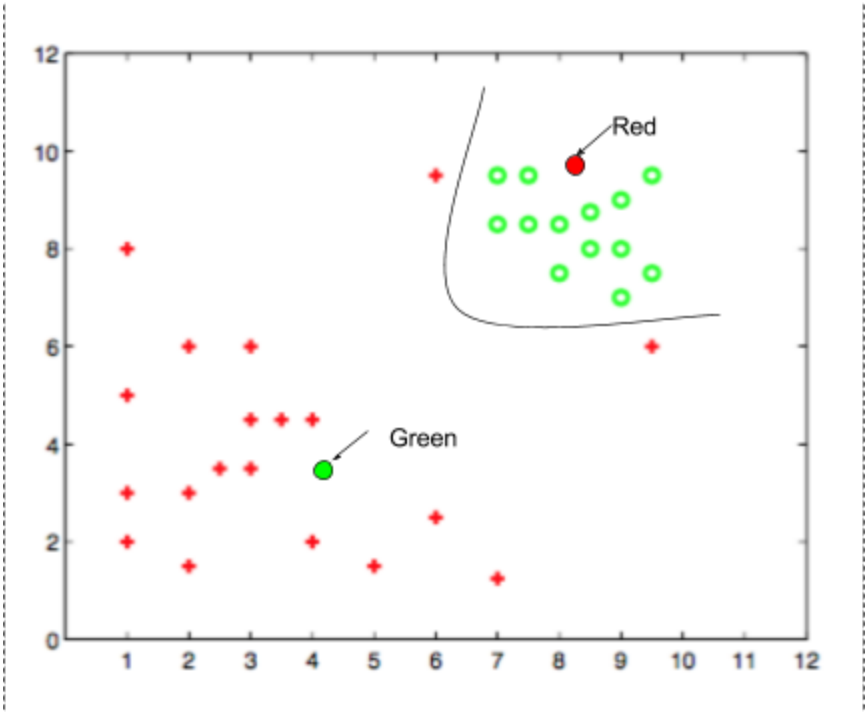
d)

For large Values of C, the curve is trying to fit the data points more precisely as discussed above. So any point which lies to the further side of the decision boundary on either side can be considered as a data point as they do not affect the curve or any change in the curve due to inclusion of it.
So, the data points included in the circles are the new entries made in the graph which do not affect the curve



e)

For large Values of C, as the curve overfits the data points, any data point which is an outlier or if it lies outside the decision boundary of its particular class will definitely change the curve. So , from the figure we can see the circles drawn will definitely affect the curve

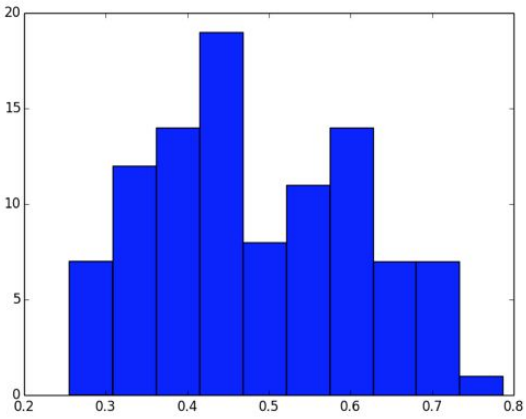


Programming Questions :

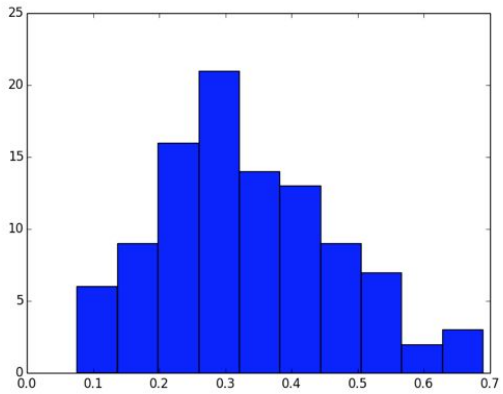
1) Bias and Variance TradeOff :

The following are the histograms for the hypothesis functions of sample size : 10

$g_1(x) = 1$

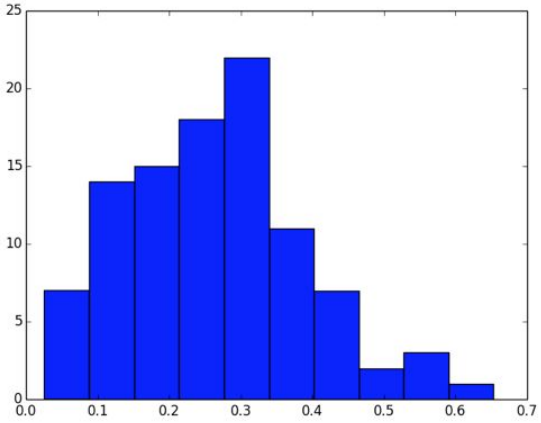


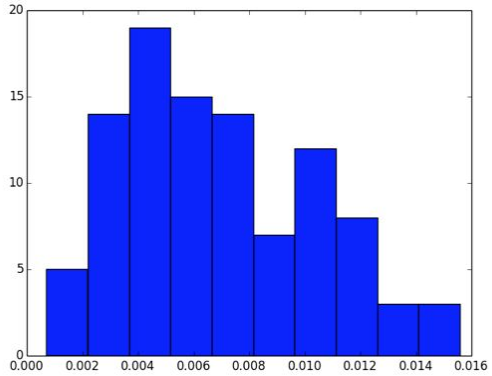
$g_2(x) = w_o$



$g_2(x) = w_o + w_1 x$

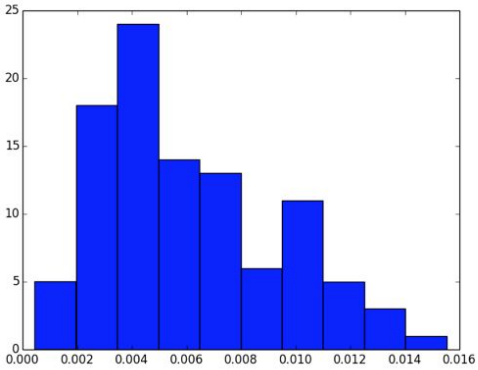
$g_2(x) = w_o + w_1 x + w_2 x^2$

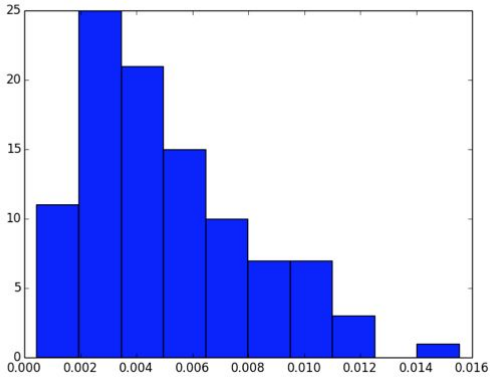




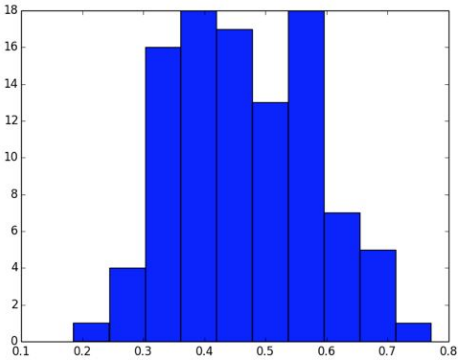
$g_2(x) = w_o + w_1 x + w_2 x^2 + w_3 x^3$

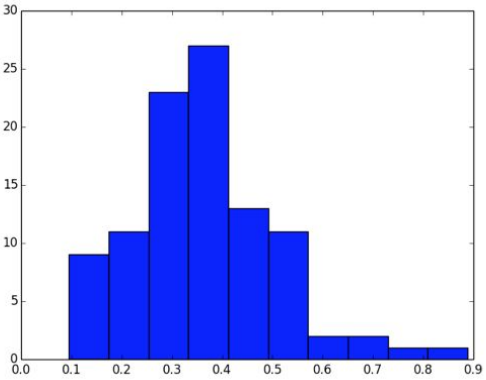
$g_2(x) = w_o + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$

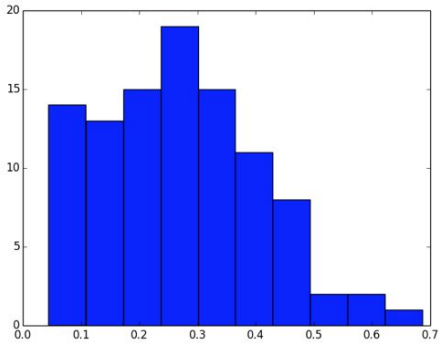


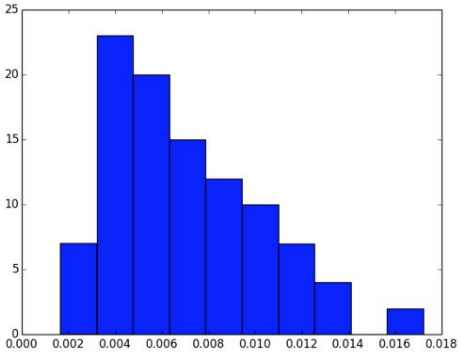


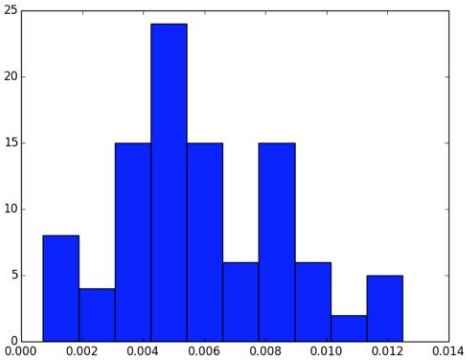
The following are the histograms for the hypothesis functions of sample size : 100 : From G1 to G6 from left to right are

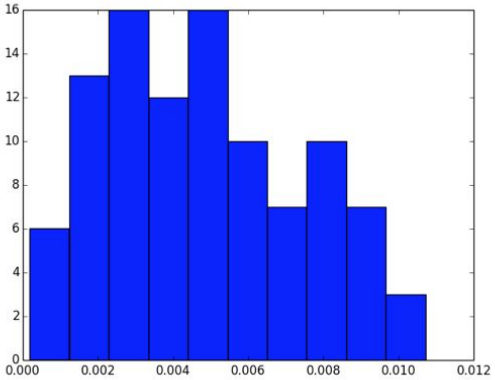












The Bias and Variance of all the functions on sample size 10 and 100 are as follows :

For Data Set of 10

Variance for G : 1 : 0.0
Bias for G : 1 ; 0.477885109133
g == 1 MSE : 0.477885109133

Variance for G : 2 : 0.0633344171781
Bias for G : 2 ; 0.379760722868
g == 2 MSE : 0.337540252423

Variance for G : 3 : 0.074605592318
Bias for G : 3 ; 0.367361530499
g == 3 MSE : 0.275173264849

Variance for G : 4 : 0.145351275348
Bias for G : 4 ; 0.36302181317
g == 4 MSE : 0.00729336758818

Variance for G : 5 : 0.187404147865
Bias for G : 5 ; 0.361013144006
g == 5 MSE : 0.00629024792276

Variance for G : 6 : 0.215345433657
Bias for G : 6 ; 0.359922015078
g == 6 MSE : 0.00536612015482

For Data Set of 100

Variance for G : 1 : 0.195401160187
Bias for G : 1 ; 0.364673454416
g == 1 MSE : 0.45986310529

Variance for G : 2 : 0.179092342381
Bias for G : 2 ; 0.360819796941
g == 2 MSE : 0.340059975213

Variance for G : 3 : 0.169966140267
Bias for G : 3 ; 0.359318882728
g == 3 MSE : 0.257278354173

Variance for G : 4 : 0.187600329244
Bias for G : 4 ; 0.358200674621
g == 4 MSE : 0.00736741841455

Variance for G : 5 : 0.202070486948
Bias for G : 5 ; 0.357340337913
g == 5 MSE : 0.00649652696232

Variance for G : 6 : 0.214190091088
Bias for G : 6 ; 0.356660901231
g == 6 MSE : 0.00545771120162

C) ***Discussion*** :

As the model complexity increases , the variance increases and the Bias decreases. That is the model ends up overfitting the data and hence the generalisation error may occur due to high variance or low bias.

However as the sample size increases. The accuracy improves as the data is more now. Increasing the sample size reduces the variance and also helps improve the Bias term.

d)

Lambda Value 0.001 ----------
MSE : 0.00728117269643
Variance for G: 2 : 0.228841466748
Bias for G: 2 : 0.356854660933
Lambda Value 0.003 ----------
MSE : 0.00728869119381
Variance for G: 2 : 0.237282956271
Bias for G: 2 : 0.356579572864
Lambda Value 0.01 ----------
MSE : 0.00736901431775
Variance for G: 2 : 0.244220191481
Bias for G: 2 : 0.356357645847
Lambda Value 0.03 ----------
MSE : 0.00797152118399
Variance for G: 2 : 0.249385732501
Bias for G: 2 : 0.356176014846
Lambda Value 0.1 ----------
MSE : 0.0125973567429
Variance for G: 2 : 0.251598318228
Bias for G: 2 : 0.356025483516
Lambda Value 0.3 ----------
MSE : 0.0322951764798
Variance for G: 2 : 0.249726699113
Bias for G: 2 : 0.355899336797
Lambda Value 1.0 ----------
MSE : 0.0925807247085
Variance for G: 2 : 0.243261196557
Bias for G: 2 : 0.355792578834

As we can see that as the Value of Lambda increases the variance decreases. That is the overfitting is reduced and the curve becomes smoother and smoother as we increase the lambda hence it is a very important parameter in overcoming the overfitting. Also as the value of lambda increases the value of the Bias term first dips and then increases.

Hence we choose the model with the parameters Lambda such that the sum of variance and bias is least and the we call it a better model to predict our data.

2) Linear and Kernel SVM

Linear SVM :

Cross Validation Accuracy :

For C = 4^{-6} Cross Validation Accuracy = 58.5%
For C = 4^{-5} Cross Validation Accuracy = 89.95%
For C = 4^{-4} Cross Validation Accuracy = 90.35%
For C = 4^{-3} Cross Validation Accuracy = 92.8%
For C = 4^{-2} Cross Validation Accuracy = 93.85%
For C = 4^{-1} Cross Validation Accuracy = 94.2%
For C = 4^{0} Cross Validation Accuracy = 94.35%
For C = 4^{1} Cross Validation Accuracy = 94.55%
For C = 4^{2} Cross Validation Accuracy = 94.5%

Average Training Time : 0.618256674873 seconds

For C = [4^{-6} $\ldots.4^{7}$] in the order
Kernel SVM in LIBSVM :

Polynomial kernel with Degree 1 :
Cross Validation Accuracy = 73.05%
Cross Validation Accuracy = 90.75%
Cross Validation Accuracy = 90.45%
Cross Validation Accuracy = 93.2%
Cross Validation Accuracy = 93.8%
Cross Validation Accuracy = 94.1%
Cross Validation Accuracy = 94.25%
Cross Validation Accuracy = 94.35%
Cross Validation Accuracy = 94.6%
Cross Validation Accuracy = 94.6%

Polynomial kernel with Degree 2

Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 90.9%
Cross Validation Accuracy = 90.7%
Cross Validation Accuracy = 92.65%
Cross Validation Accuracy = 95.25%
Cross Validation Accuracy = 95.8%
Cross Validation Accuracy = 95.9%
Cross Validation Accuracy = 96.15%
Cross Validation Accuracy = 96.3%

Cross Validation Accuracy = 96.45%
Cross Validation Accuracy = 96.75%

Polynomial kernel with Degree 3

Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 85.4%
Cross Validation Accuracy = 91.7%
Cross Validation Accuracy = 92.4%
Cross Validation Accuracy = 95.25%
Cross Validation Accuracy = 96%
Cross Validation Accuracy = 96.25%
Cross Validation Accuracy = 96.3%
Cross Validation Accuracy = 96.2%
Cross Validation Accuracy = 96.35%
Cross Validation Accuracy = 97.05%

Average Training Time : 1.03705484779 seconds

RBF Kernel with C = $[4^{-6}$ ***....*** $4^{7}]$ ***and G =*** $[4^{-7}$ ***....*** $4^{-1}]$ ***:***

In order of the for loop of G inside C

Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 80.65%
Cross Validation Accuracy = 90.1%
Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 87.05%
Cross Validation Accuracy = 90.85%
Cross Validation Accuracy = 91.15%
Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 88.1%
Cross Validation Accuracy = 90.7%
Cross Validation Accuracy = 90.85%
Cross Validation Accuracy = 93.4%

Cross Validation Accuracy = 55.75%
Cross Validation Accuracy = 87.85%
Cross Validation Accuracy = 90.65%
Cross Validation Accuracy = 90.95%
Cross Validation Accuracy = 94.25%
Cross Validation Accuracy = 96%
Cross Validation Accuracy = 88.05%
Cross Validation Accuracy = 90.7%
Cross Validation Accuracy = 91.2%
Cross Validation Accuracy = 93.45%
Cross Validation Accuracy = 94.95%
Cross Validation Accuracy = 96.55%
Cross Validation Accuracy = 90.6%
Cross Validation Accuracy = 91.2%
Cross Validation Accuracy = 93.9%
Cross Validation Accuracy = 94.2%
Cross Validation Accuracy = 95.85%
Cross Validation Accuracy = 97.1%
Cross Validation Accuracy = 91.1%
Cross Validation Accuracy = 93.6%
Cross Validation Accuracy = 94.65%
Cross Validation Accuracy = 95%
Cross Validation Accuracy = 96.9%
Cross Validation Accuracy = 96.6%
Cross Validation Accuracy = 93.85%
Cross Validation Accuracy = 94.25%
Cross Validation Accuracy = 94.15%
Cross Validation Accuracy = 95.65%
Cross Validation Accuracy = 96.8%
Cross Validation Accuracy = 96.05%
Cross Validation Accuracy = 94.55%
Cross Validation Accuracy = 94.35%
Cross Validation Accuracy = 95%
Cross Validation Accuracy = 96.35%
Cross Validation Accuracy = 96.85%
Cross Validation Accuracy = 96.9%
Cross Validation Accuracy = 93.65%
Cross Validation Accuracy = 94.65%
Cross Validation Accuracy = 95.95%
Cross Validation Accuracy = 96.35%
Cross Validation Accuracy = 96.5%
Cross Validation Accuracy = 96.9%
Cross Validation Accuracy = 94.8%

Cross Validation Accuracy = 95.15%
Cross Validation Accuracy = 96.25%
Cross Validation Accuracy = 96.3%
Cross Validation Accuracy = 96.55%
Cross Validation Accuracy = 96.35%

Average Training Time : 0.787077431168 seconds

Based on the RBF Function and Polynomial kernel we choose RBF as the kernel type as it gives the best result for training errors :

The parameters chosen for RBF are Gamma = 4^{-1}
The parameters chosen for RBF are C = 4^{5}

And the testing Accuracy with this kernel and parameters : 95.3%

optimization finished, #iter = 1505
nu = 0.013330
obj = -26961.705834, rho = -0.227089
nSV = 707, nBSV = 23
Total nSV = 707
<svm.svm_model object at 0x10338b4d0>
Accuracy = 95.3% (1906/2000) (classification)