CSCI 567 Assignment 3
Fall 2016

Snehal Adsule
2080872073
adsule@usc.edu

October 17,2016

1 Problem 1

1.1 1 (a)

Closed Form Given that $\hat{\beta}_\lambda = argmin_\beta \{ \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \mid\mid \beta \mid\mid_2^2 \}$
Differentiating wrt β

$$\frac{\delta \hat{\beta}_\lambda}{\delta \beta} = \frac{2}{n} \{ \sum_{i=1}^{n} (y_i - x_i^T \beta)(-x_i^T) + \lambda \beta \} = 0$$
$$=> \frac{2}{n} \{ -X^T Y + X^T \beta X + \lambda \beta \} = 0$$
$$=> \beta (X^T X + \lambda) = Y X^T$$
$$=> \hat{\beta} = (X^T X + \lambda)^{-1} X^T . Y$$

using $Y = X\beta^* + \epsilon$
$\hat{\beta} = (X^T X + \lambda)^{-1} X^T . (X\beta^* + \epsilon)$
The guassian distribution for the noise is, $\epsilon N(0, \sigma^2)$.
Using affine transformation distribution of y can be written as

Thus, $\hat{\beta} = (X^T X + \lambda)^{-1} X^T . (X\beta^* + \epsilon)$
And, $Y N(X\beta^*, \sigma^2 I)$
$=> \hat{\beta}_\lambda = ((X^T X + \lambda)^{-1} X^T X \beta^*, (X^T X + \lambda)^{-1} X^T X (X^T X + \lambda I)^{-1})$

1.2 1 (b)

Bias Term

$$\begin{aligned} & E[x^T\hat{\beta}_\lambda] - x^T\beta^* \\ = x^T(E[\hat{\beta}_\lambda] - \beta^*) &= x^T((X^TX+)^{-1}X^TX\beta^*\beta^*) \\ &= x^T((X^TX+\lambda)^{-1}X^TX - I)\beta^* \end{aligned}$$

next

1.3 1 (c)

Variance Term

$$\begin{aligned} E[(x^T(\beta_\lambda E[\beta_\lambda]))^2] &= x^T(X^TX+\lambda)^{-1}X^TX(XX^T+\lambda I)^{-1}x \\ &= ||X(XX^T+\lambda)^{-1}x||_2^2 \end{aligned}$$

1.4 1 (d)

We can observe that , with Part b. and Part c. of the bias and variance tradeoff if λ increases, the bias term also increases while the variance term decreases. And when λ is small, the bias term si expected to be smaller and the variance term will be larger, comparatively.

2 Kernel Construction

2.1 2. (a)

To prove that , $k_3(x,x') = a_1k_1(x,x') + a_2k_2(x,x')$ where$a_1, a_2 \geq 0$

Since $k_1(x,x')$ is positive definite, $\forall y \in \mathbf{R}$,

$$\begin{aligned} & y^TK^{(1)}y \geq 0 \\ & \quad \text{where } K_{ij}^{(1)} = k_1(x_i, x_j') \end{aligned}$$

Similarly,

$$\begin{aligned} & y^TK^{(2)}y \geq 0 \\ & \quad \text{where } K_{ij}^{(2)} = k_2(x_i, x_j') \end{aligned}$$

Adding ,the above two equations, we get

$$\begin{aligned} y^T(K^{(1)} + K^{(2)})y \geq 0 \ \forall y \in \mathbf{R} \implies \\ y^TK^{(3)}y \geq 0 \ \forall y \in \mathbf{R} \\ \text{where} K_{ij}^{(3)} = k_3(x_i, x_j') \end{aligned}$$

2.2 2. (b)

To prove , $k_4(x, x') = f(x)f(x')$ $K^{(4)}_{ij} = k_4(x_i, x_j) = f(x_i)f(x'_j)$
Since $f(x)$ is a real valued function, consider $K^{(4)}$

$$K^{(4)} = \begin{bmatrix} f(x_1)f(x'_1) & f(x_1)f(x'_2) & \cdots & f(x_1)f(x'_n) \\ \vdots & & & \\ f(x_n)f(x'_1) & f(x_n)f(x'_2) & \cdots & f(x_n)f(x'_n) \end{bmatrix}$$

$$K^{(4)} = F(\vec{x})_{n\times 1} F(\vec{x})^T_{1\times n}$$

where

$$F(x)^T_{1\times n} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots f(x_n) \end{pmatrix}$$

Therefore, $y^T K^{(4)} y = y^T F(x) F(x)^T y = y^T F(x)(y^T F(x))^T = ||y^T F(x)||_2^2 \geq 0$
We can say , $k_2(.,.)$ is a valid kernel function!.

2.3 2. (c)

To prove that $k_5(x, x') = k_1(x, x')k_2(x, x')$ $K^{(5)} = K^{(1)} \circ K^{(2)}$ where $\circ$ denotes the Hadamard product. Using the Schur product for$K^{(1)}, K^{(2)}$ we can prove this.

Since, k_1 and k_2 are valid kernel function $\exists v_i w_j$ the eigen vectors of matrix K_1 and K_2 defines such that:
$K^{(1)} = \sum_i \lambda_i v_i v_i^T$ and $K^{(2)} = \sum_j \mu_j w_j {w_j}^T$
Now,

$$\begin{aligned} K^{(5)} &= K^{(1)} \circ K^{(2)} \\ &= \sum_i \lambda_i v_i v_i^T \circ \sum_j \mu_j w_j {w_j}^T \\ &= \sum_{i,j} \lambda_i \mu_j (v_i v_i^T) \circ w_j {w_j}^T \\ &= \sum_{i,j} \lambda_i \mu_j (v_i \circ w_j)(v_j \circ w_j)^T \\ &\geq 0 \end{aligned}$$

As, $(v_i \circ w_j)(v_j \circ w_j)^T = ||v_i w_j||_2^2 \geq 0$

3 Kernel Regression

3.1 3.a

Given that ,$min_w(\sum_i (y_i - w^T x_i)^2 + \lambda ||w||_2^2)$
We can think of it as vector and rewrite is as ,

$min_w(||y - w^T X||_2^2 + \lambda||w||_2^2)$

$$
\begin{aligned}
f(w) &= min_w(||y - Xw||_2^2 + \lambda||w||_2^2) \\
&= (y - Xw)^T(y - Xw) + \lambda w^T w \\
&= (y^T - w^T X^T)(y - Xw) + \lambda w^T w \\
&= y^T y - y^T Xw - w^T X^T y + w^T X^T Xw + \lambda w^T w \\
&= y^T y - (X^T y)^T w - w^T X^T y + w^T X^T Xw + \lambda w^T w \\
\frac{\partial f(w)}{\partial w} &= -X^T y - X^T y + 2\lambda w + (X^T Xw + (XX^T w)) = 0 \\
&= 2\lambda w + 2X^T Xw - 2X^T y = 0 \\
\text{w}(\lambda I_D + X^T w) &= X^T y
\end{aligned}
$$

=> w* = $(X^T Xw + \lambda I_D)^{-1} X^T y$, where I_D denotes DxD identity matrix

3.2 3.b

After applying the non linear feature mapping , the solution should be similar
$min_w(||y - w^T \Phi||_2^2 + \lambda||w||_2^2)$

=> w = $(\Phi^T \Phi + \lambda I_D)^{-1} \Phi^T y$
Using the identity:

$$(P^{-1} + B^T R^{-1} B)^{-1} B^T R^{-1} = PB^T (BPB^T + R)^{-1}$$

and assuming matrix inversion is valid

$$\big((\lambda I_D + \Phi^T \Phi)^{-1}\big)\Phi^T y = \Phi^T \big(\Phi\Phi^T + \lambda I_N\big)^{-1} y$$

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

3.3 3.c

$$\hat{y} = w^{*T} \Phi(x)$$

can be written as

$$\hat{y} = \big(\Phi^T (\Phi\Phi^T + \lambda I_N)^{-1} y\big)^T \Phi(x) = y^T \big((\Phi\Phi^T + \lambda I_N)^{-1}\big)^T \Phi^T \Phi(x)$$

$$
\begin{aligned}
\hat{y} &= y^T\big((\Phi\Phi^T + \lambda I_N)^{-1}\big)^T\Phi^T\Phi(x) \\
&= y^T\big((\Phi\Phi^T + \lambda I_N)^T\big)^{-1}\Phi^T\Phi(x), Using, (A^{-1})^T = (A^T)^{-1} \\
&= y^T\big((\Phi^T\Phi + \lambda I_N)\big)^{-1}\Phi^T\Phi(x) \\
&= y^T(K + \lambda I_N)^{-1}\kappa(x)
\end{aligned}
$$

Where $K_{ij} = \Phi_i^T\Phi_j$ and $\kappa(x) = \phi^T\phi^T(x)$ (given)

3.4 3.d

We can say that kernel ridge regression is $O(n^3)$ for n data points, considering the multiplication and inversion of matrices. However, linear regression can be presented as quadratic programing and hence is $O(n^2)$. Kernel $N \times N$ compared to $D \times D$(for ridge regression without kernel) as in Part (b). In cases where $d < n$ this leads to an extra operations for computing K .

Problem 4

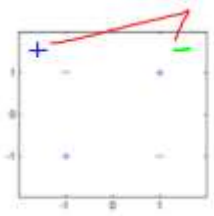
4 (a) No, it cannot be linearly separable for the XOR, in the original feature space.

4 (b) $y = w^T \emptyset(x)$, we can calculate the w^T by matrix multiplication of X^TY as [0,0,0,1], also by intuition we can see x1x2 represent the y completely and other terms don't contribute much and can be eliminated.

w^T= [0,0,0,1]

Y	1	X1	X2	x1x2
1	1	1	1	**1**
-1	1	1	-1	**-1**
-1	1	-1	1	**-1**
1	1	-1	-1	**1**

4 (c) We can draw the opposite class points near the current point and then it will be difficult to separate the space linearly in the feature space defined as below.



4 (d) K(x,x) corresponds to Kernel function given by the inner product of feature vector in another vector space.

$$K(x,x) = \phi(x)\phi'(x)$$

$$= [1, x1, x2, x1x2]\ [1, x1', x2', x1'x2']^T$$

$$= 1 + x1.x1' + x2.x2' + x1.x2.x1'.x2'$$

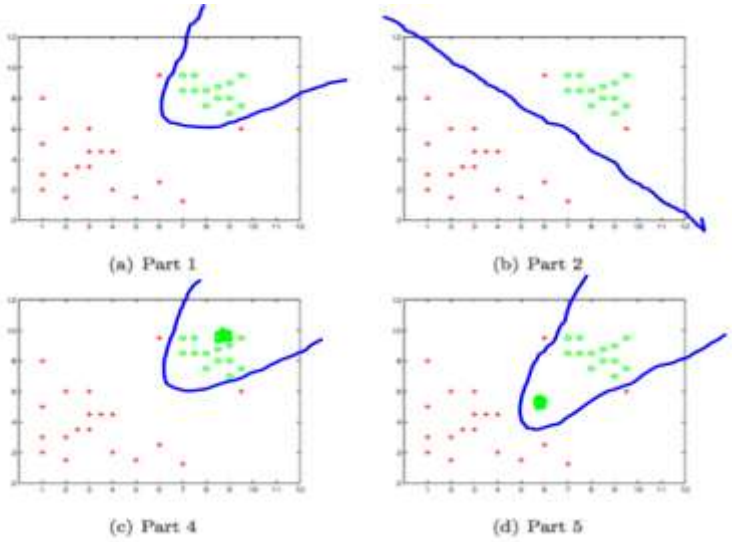
Problem 5

Consider the solution $\min_{(w,b)} = \frac{1}{2}||w||^2 + C\sum\xi$

5 (a) For large values of C $\rightarrow \infty$, we are penalizing less with (1/ λ) but shrinking the margin heavily. But we are penalizing more for misclassified points, and the decision boundary will separate the data better as much as possible. Please find the figure below.

5 (b) For C -> 0, we not penalizing more for misclassified and maximizing the margins, we can have some misclassified points. Please find the drawing below.

5 (c) As the sensor data is not reliable and we have to account for the errors or misclassification, we can use C approaching 0 as in 5(b) as better classification.



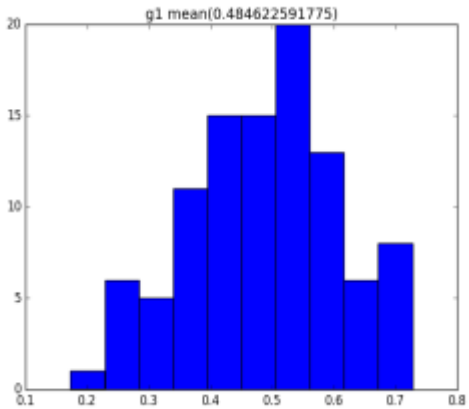
5 (d) If the point lies close to the positive examples, even large C won't change the decision boundary as shown above.

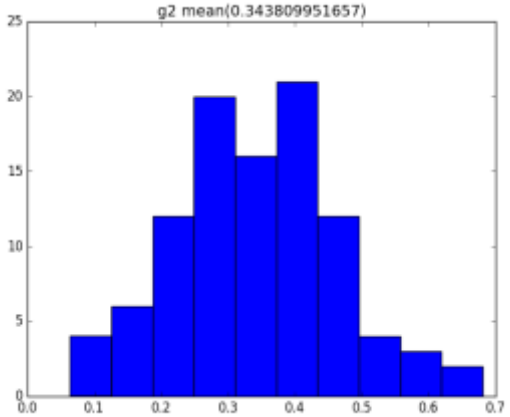
5 (e) If we add a point closer to the opposite classes, which will result in misclassification with the initial approach can result in affecting the decision boundary learned for large values of C as shown in the figure above.

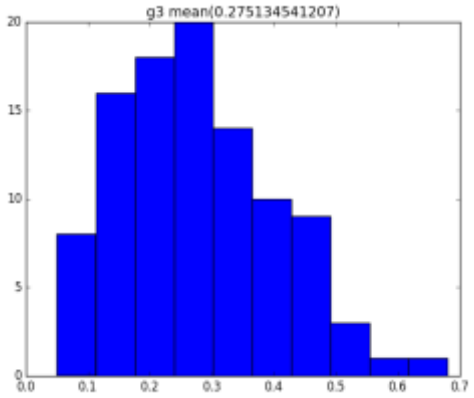
Programming

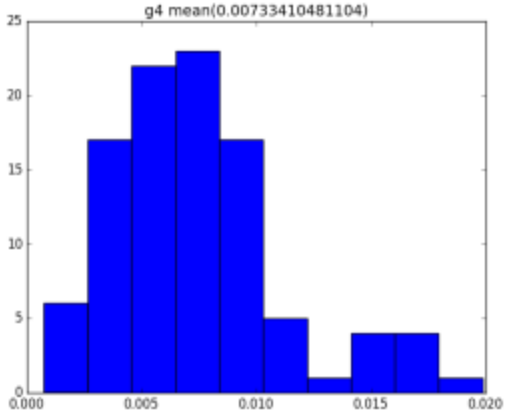
6.1 (a) 100 datasets with 10 samples - Computed the bias^2 and variance for the various g(x) function, results as follows.

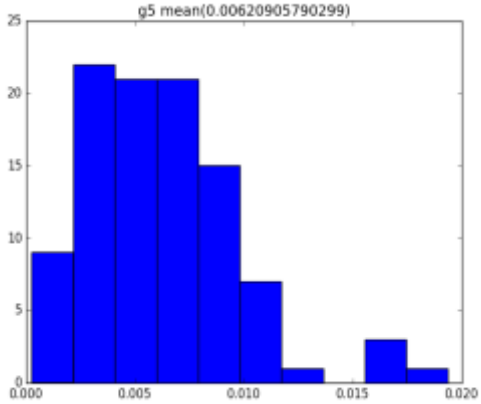
	MSE	BIAS^2	VAR
g1	0.484623	0.470979	0.0
g2	0.343810	0.366203	0.034857
g3	0.275135	0.369681	0.10187
g4	0.007334	0.169967	0.173902
g5	0.006209	0.16999	0.174956
g6	0.004937	0.169984	0.175693

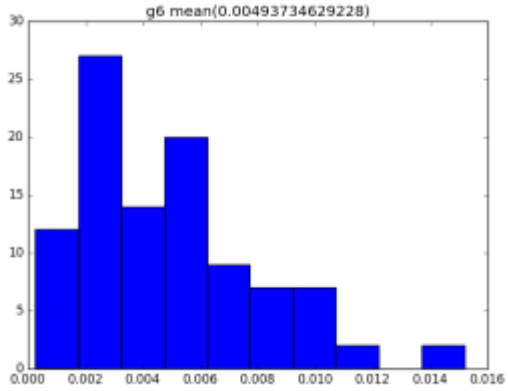






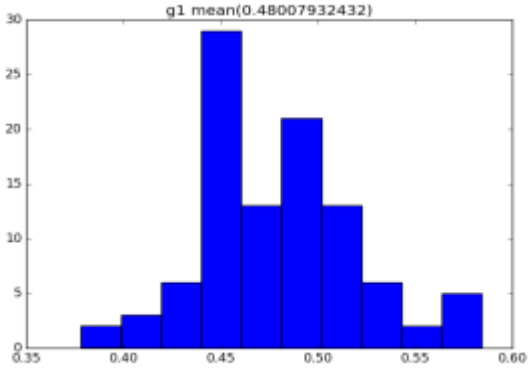


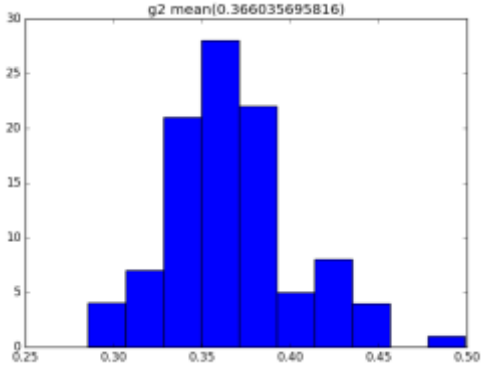


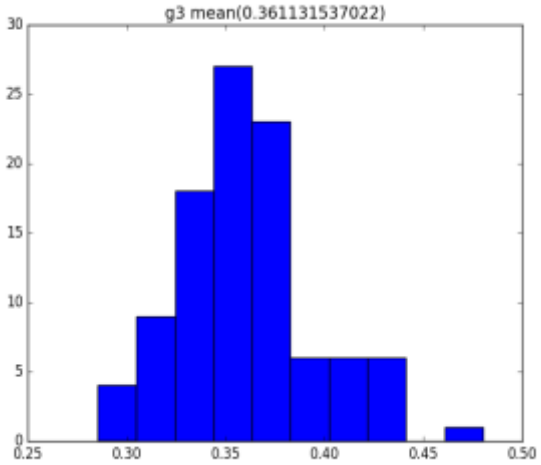


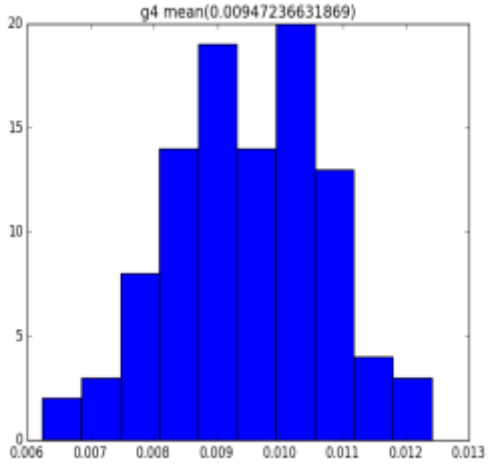
6 .1(b) 100 datasets with 100 samples - computed bias^2 and variance as follows:

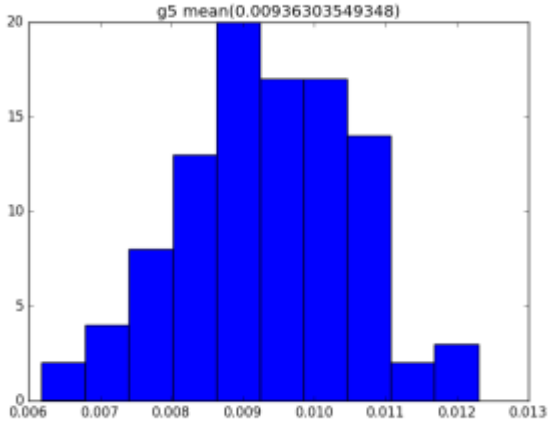
	MSE	$BIAS^2$	VAR
g1	0.480079	0.468741	0.0
g2	0.366036	0.358126	0.003396
g3	0.361132	0.358591	0.008276
g4	0.009472	0.018377	0.01897
g5	0.009363	0.018377	0.018984
g6	0.009241	0.018379	0.018996

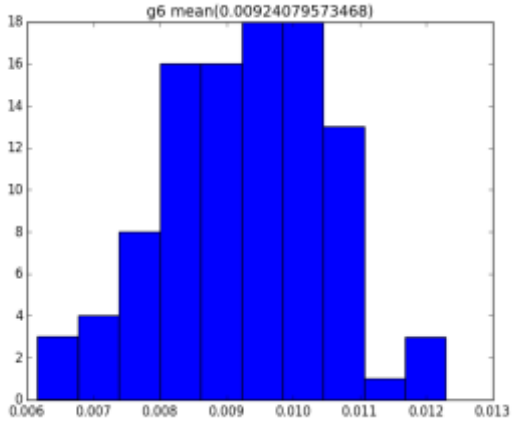












6 1.(c) It can be observed that with the increase in the model complexity the squared bias decreases and the variance increases and the mean squared error decreases, as we try to fit the model better for each point.
The impact of sample size increase was seen more significant on the variance, with decrease in the variance for larger sample size, as 6(b) compared to 6(a). Though bias also varied with larger sample size for the corresponding hypothesizes, but not much increase. Otherwise, the bias and variance followed the tradeoff pattern same as with the complexity. Small sample size is source for the variance.

6 1.(d)

Lambda	MSE	$BIAS^2$	VAR
0.001	0.000206	0.021334	0.02142
0.003	0.000184	0.020867	0.021087
0.01	0.000202	0.019087	0.019272
0.03	0.000184	0.021346	0.02148
0.1	0.000224	0.021106	0.021198
0.3	0.000608	0.023374	0.022039
1	0.003937	0.024233	0.017955

It is observed that as λ the increases, the variance seems to be decreasing along with the bias increases. With higher λ we try to give large penalty and hence the coefficient tend to be close to zero, at the cost of the bias increase. With the regularization we are trying to reduce the variance and thus the mean squared error, this is typically useful when we have more number of parameters than the number of samples.

6.2 LIBSVM

Linear LibSVM with 3-fold CV

Training time =22.6619999409 seconds

Cross Validation Accuracy :-

C	CV Accuracy
{0.000244140625	55.75
0.000977	88.5
4	94.4
0.25	93.8
1	93.85
16	94.55
0.003906	91.25
0.015625	92.55
0.0625	94.1

Polynomial LibSVM with 3 fold

Training time = 160.46600008 seconds

Cross Validation Accuracy :-

Degree	C	CV accuracy
1	0.015625	55.75
1	0.0625	89.5
1	0.25	91.15
1	1	93.35
1	4	94.15
1	16	94.4
1	64	94
1	256	94.95
1	1024	94.5
1	4096	95
1	16384	94.35
2	0.015625	55.75
2	0.0625	88.75
2	0.25	91.85
2	1	93.05
2	4	94.6
2	16	95.8
2	64	97
2	256	96.3
2	1024	96.25

2	4096	96
2	16384	95.9
3	0.015625	55.75
3	0.0625	72.15
3	0.25	91.65
3	1	92.95
3	4	95.25
3	16	96.05
3	64	96.85
3	256	96.85
3	1024	96.55
3	4096	96.95
3	16384	96.25

RBF LibSVM 3-fold

training time = 185.242000103 seconds

Cross Validation Accuracy :-

Gamma	C	CV accuracy
6.10E-05	0.015625	55.75
6.10E-05	0.0625	55.75
6.10E-05	0.25	55.75
6.10E-05	1	55.75
6.10E-05	4	67.8
6.10E-05	16	91
6.10E-05	64	91.35
6.10E-05	256	93.85
6.10E-05	1024	94.35
6.10E-05	4096	94.6
6.10E-05	16384	94.75
0.000244141	0.015625	55.75
0.000244141	0.0625	55.75
0.000244141	0.25	55.75
0.000244141	1	67.1
0.000244141	4	90.6
0.000244141	16	91.25
0.000244141	64	93.4
0.000244141	256	94.35
0.000244141	1024	94.45
0.000244141	4096	94.4
0.000244141	16384	94.55
0.000976563	0.015625	55.75
0.000976563	0.0625	55.75
0.000976563	0.25	67

0.000976563	1	90.55
0.000976563	4	91.3
0.000976563	16	93.95
0.000976563	64	94.3
0.000976563	256	94.8
0.000976563	1024	94.65
0.000976563	4096	95.25
0.000976563	16384	96.6
0.00390625	0.015625	55.75
0.00390625	0.0625	64.25
0.00390625	0.25	90.8
0.00390625	1	91.15
0.00390625	4	93.35
0.00390625	16	94.7
0.00390625	64	94.95
0.00390625	256	95.65
0.00390625	1024	96.25
0.00390625	4096	97
0.00390625	16384	97
0.015625	0.015625	56.1
0.015625	0.0625	90.4
0.015625	0.25	91.2
0.015625	1	93.65
0.015625	4	94.45
0.015625	16	96.15
0.015625	64	96.7
0.015625	256	96.8
0.015625	1024	96.45
0.015625	4096	96.45
0.015625	16384	96.5
0.0625	0.015625	87.6
0.0625	0.0625	91.85
0.0625	0.25	93
0.0625	1	95.75
0.0625	4	97.1
0.0625	16	97
0.0625	64	96.9
0.0625	256	96.25
0.0625	1024	96.5
0.0625	4096	95.8
0.0625	16384	96.1
0.25	0.015625	60.55
0.25	0.0625	92

0.25	0.25	96
0.25	1	97.55
0.25	4	97.45
0.25	16	97.15
0.25	64	97
0.25	256	96.75
0.25	1024	96.5
0.25	4096	96.95
0.25	16384	97.15

6.2. (d) Best kernel and hyperparameters

Linear (C= 16.0) best accuracy =94.55
Polynomial (degree =2, C=64.0) best accuracy=97.0
RBF ((gamma=0.25, C=1.0)) best accuracy =97.55
Best Kernel Type RBF against hyper parameters are (gamma=0.25, C=1.0) for 3 fold CV.
Though, overall average best accuracy is for polynomial libSVM .

P**redicting for the best parameter**
Using Kernel Type RBF against hyper parameters are (gamma=0.25, C=1.0)
Training Accuracy = 98.6% (1972/2000) (classification)
ACC=98.6 MSE=0.056 SSC=0.94421556104
Test Accuracy = 95.55% (1911/2000) (classification)
ACC=95.55 MSE=0.178 SSC=0.828770507909