ML Assignment3

Arpita Agrawal, USC ID: 8100884538

October 2016

1 Bias Variance Trade-off

a) $Given: y = x^T\beta^* + \epsilon, \quad \beta_\lambda = \arg\, min_\beta \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda||\beta||^2$ - (1)

Using Ridge Regression we already know the analytical solution for $\beta = (X^TX + \lambda I)^{-1}X^TY$, also similar derivation has been made in question 3(a) below

Substituting $y = x^T\beta^* + \epsilon$, where ϵ is normaly distributed in the above equation, assumed to be distributed according standard normal distribution $\mathcal{N}(0, 1)$

Applying Affine transformation $Ax + b = \mathcal{N}(A\mu + b, A^T\sigma A)$on equation (1)

$$(X^TX + \lambda I^{-1})X^T(X^T\beta^* + \epsilon) = (X^TX + \lambda I)^{-1}(XX^T\beta^* + X^T\epsilon) = (X^TX + \lambda I)^{-1}\mathcal{N}(XX^T\beta^*, XX^T) \quad -(2)$$

Applying Affine transformation again on equation (2)

Using Transpose and Inverse properties of Linear Algebra

$$\mathcal{N}((X^TX + \lambda I)^{-1}XX^T\beta^*, (X^TX + \lambda I)^{-1}XX^T((X^TX + \lambda I)^{-1}))^T$$

$$= \mathcal{N}((X^TX + \lambda I)^{-1}XX^T\beta^*, (X^TX + \lambda I)^{-1}XX^T(X^TX + \lambda I)^{-1})$$

b)

$$E[X^T\hat{\beta_\lambda}] - X^T\beta^* = X^T(E[\hat{\beta_\lambda}] - \beta^*)$$

Using the μ value written in the a) part in Normal distribution form,

$$X^T((X^TX + \lambda I)^{-1}XX^T\beta^* - \beta^*) = X^T((X^TX + \lambda I)^{-1}XX^T - 1)\beta^*$$

Bias term $= X^T((X^TX + \lambda I)^{-1}XX^T - 1)\beta^*$

c)

Variance $= E[(X^T\hat{\beta_\lambda} - E[X^T\hat{\beta_\lambda}])^2] = XE[(\hat{\beta_\lambda} - E[\hat{\beta_\lambda}])^2]X^T = X(\sigma^2)X^T$

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

d) We know,square error can be decomposed into- bias square + variance + noise

$X^T((X^TX + \lambda I)^{-1}XX^T - 1)\beta^* + X((X^TX + \lambda I)^{-1}XX^T(X^TX + \lambda I)^{-1}))X^T$ + Noise

As lambda increases Variance decreases, as the second term is a product of inverse of lambda, so when lamda increase the entire term decreases. But, Bias increases as there is one more term in addition with the term in product with lambda inverse term.

When Lamda decreases Variance increases and Bias, decreases. So, lamda and variance are inversly related and bias increases wid Lambda increases and so on.

2 Kernel Construction

(a) $k_3(x, x^{'}) = a_1k_1(x, x^{'})) + a_2k_2(x, x^{'})$ where $a_1, a_2 \geq 0$

We know, for every positive semi-definite matrix $\mathbf{A}$ $\mathrm{R}^{n\times n}$ and arbitrary vector $\mathbf{x}$ $\mathrm{R}^{n\times 1}$,we have $x > \mathbf{Ax} \geq 0$ Since, k_1, k_2 are kernel functions, lets say we have a vector $\mathbf{Y}$, we can write,

$a_1Yk_1(x, x^{'})Y^T \geq 0$ and $a_2Yk_2(x, x^{'})Y^T \geq 0$

adding the above two equations,

$a_1Yk_1(x, x^{'})Y^T + a_2Yk_2(x, x^{'})Y^T \geq 0$

$Y(a_1k_1(x, x^{'}) + a_2k_2(x, x^{'}))Y^T \geq 0$

Therefore for, $a_1, a_2 \geq 0$, $Y(a_1k_1 + a_2k_2)Y_1 \geq 0$, So, our function $k_3(x, x\prime)$ is also positive semi-definite and addition of two symmetric matrices will also give a symmetric matrix hence is a valid kernel function.

(b) $k_4(x, x^{'}) = f(x)f(x^{'})$ where f(.) is a real valued function

Now, we know $f(x)f(x^{'}) = f(x^{'})f(x)$. So, if we make a matrix of $f(x)f(x^{'})$

$$k_4 = \begin{vmatrix} f(x_1)f(x_1) & f(x_1)f(x_2) & . & . & f(x_1)f(x_n) \\ f(x_2)f(x_1) & f(x_2)f(x_2) & . & . & f(x_2)f(x_n) \\ f(x_2)f(x_1) & & & & \\ f(x_4)f(x_1) & & & & \\ & & & & \\ f(x_n)f(x_1) & f(x_n)f(x_2) & . & . & f(x_n)f(x_n) \end{vmatrix}.$$

The above matrix is a symmetric matrix

Now, the above matrix can be written in the form YY^T, where Y = $f(x) = [f(x_1), f(x_2), f(x_3)..., f(x_n))]$

$$Y^T = f(x') = \begin{vmatrix} f(x_1) \\ f(x_2) \\ . \\ . \\ . \\ f(x_n) \end{vmatrix}$$

K_4 can be written as YY^T, for any Matrix B, we can write BYY^TB^T which can also be written as $(BY)(BY)^T = ||BY||^2$

$||BY||^2 \geq 0$, so our function k_4 is also positive semi definite and hence a valid kernel function.

(c) $k_5(x, x') = k_1(x, x')k_2(x, x')$

$$k_1(x, x')k_2(x, x') = (\Phi_1(x) \cdot \Phi_1(x'))(\Phi_2(x) \cdot \Phi_2(x'))$$

$$k_1(x, x')k_2(x, x') = \sum_i \Phi_1(x_i) \cdot \Phi_1(x'_i) \sum_j \Phi_2(x_j) \cdot \Phi_2(x'_j)$$

$$= \sum_{i,j} \phi_1(x_i)\phi_1(x'_i)\phi_2(x_j)\phi_2(x'_j) = \sum_{i,j} \phi_1(x_i)\phi_2(x_j)\phi_1(x'_i)\phi_2(x'_j) = \sum_{i,j} \phi_3(x_{ij})\phi_3(x'_{ij})$$

$= K_6(x, x') = \Phi(x)\Phi'(x)$ is also a valid positive semi definite kernel function.

3 Kernel Regression

a)

$$J = \min_w \sum_n (y_i - w^T x_i)^2 + \lambda ||w||_2^2$$

$$\frac{\partial J}{\partial w} = \frac{\partial (y_i - w^T x_i)^2 + \lambda ||w||_2^2}{\partial w}$$

Simplifying our coist function by opening the brackets, and doing vector multiplications J can be written as below,

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

$$\frac{\partial J}{\partial w} = \frac{\partial (Y^T - X^T w)(Y - w^T X) + \lambda w^T w)}{\partial w} = \frac{\partial}{\partial w}(Y^T Y - (X^T Y)w - w^T X^T Y + w^T X^T X w)$$

$$= -X^T Y - X^T Y + 2X^T X w + 2\lambda w = 2(-X^T Y + X^T X w + 2\lambda w) = -X^T Y + (X^T X + 2\lambda)w = 0$$

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

b) Using Feature transformation when X is transformed to $\Phi(X_i)$

w^*can be written as per below following the same derivation steps in the part (a)

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

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

equation (1) can be written as $w^* = \Phi(X)^T (\lambda I + \Phi(X)^T \Phi(X))^{-1} Y$

c) Prediction of transformed test feature vector $\Phi(x)$ can be written as $\hat{y} = w^{*T}\Phi(x)$

Using the value of optimal w found in b) and substuting it in the above equation,

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

Using Transpose Multiplication and Inverse properties, simplifying teh above equation

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

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

$\Phi^T\Phi$ can be written as K, where K is the kernel function, k(x) = $\Phi^T(x_i)\Phi(x_j))$

$$\hat{y} = y^T(K + \lambda I)^{-1}k(x)$$

d) Assuming Matrix multiplication and Inverting of matrices of dimension $N \times N$ takes $O(N^3)$.
So, for our closed form solution $w = (\lambda I + X^TX)^{-1}X^TY$ takes $O(D^3)for X^TX$ multiplication and inverse and another $N \times D^2$ for multiplication. In total takes $O(D^3 + ND^2)$

For Kernel Regression, $w^* = (\lambda I + \Phi(X)^T\Phi(X))^{-1}\Phi(X)^TY$, it takes $O(N^3)$ for computing transformed Kernel function multiplication $(\Phi(X)^T\Phi)$ leading to an overall $O(N^3)$ complexity.

4 Support Vector Machine

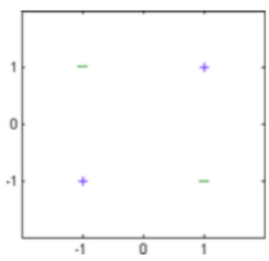

Figure 1: Original Space

1) Are the positive examples linearly separable from the negative examples in the original space. The points in the above graph **are not** linearly separable. A polynomial graph will be required in this space to divide the points into two clusters. There is no line which can divide these points into two sets with positive and negative examples clusters.

2) $\Phi_1(x) = [1, x_1, x_2, x_1x_2]$, where x_1, x_2 are features of a single training example.$\Phi_1(x)$ is the which transforms the features into a higher dimension space. so, that we can divide the points into negative and positive clusters.
After applying features transformation my vectors become:

$x_1 = (1, 1) -> $ **[1,1,1,1]**, $x_2 = (-1, -1) -> $ **[1,-1,-1,1]** positive value samples
$x_3 = (1, -1) -> $ **[1,1,-1,-1]**, $x_4 = (-1, 1) -> $ **[1,-1,1,-1]** negative value samples

If we look at the above transformed samples in the below matrix $N \times M$ form,

$$k_4 = \begin{vmatrix} 1 & 1 & 1 & \mathbf{1} \\ 1 & -1 & -1 & \mathbf{1} \\ 1 & 1 & -1 & \mathbf{-1} \\ 1 & -1 & 1 & \mathbf{-1} \end{vmatrix}.$$

we will see that, the first two rows are the examples which are samples of positive class and the bottom two rows, are examples of samples of negative examples. If, we look closely we will see that the last column defines how the samples are going to be classified as.

So, my classification depends on the product of x_1x_2, so our parameter matrix w_T can have [0 0 0 1] as one of the solutions.

If we take product of X and w_T, we get

$$\begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{vmatrix} \cdot \begin{vmatrix} 0 \\ 0 \\ 0 \\ 1 \end{vmatrix} = \begin{vmatrix} \mathbf{1} \\ \mathbf{1} \\ -1 \\ -1 \end{vmatrix}$$

y = [1 1 -1 -1], divides my 4 training examples perfectly into clusters, one with $y > 0$, other $y < 0$
cofficient vector $\mathbf{w = [0\ 0\ 0\ 1]}$

3) If i add another point (2,-2) as positive example, then we will not be able to divide the space into positive and negative clusters using the above strategy.

4) K(x, x) = $\Phi(x)\Phi(x)^T = [1 \quad x_1 \quad x_2 \quad x_1x_2][1 \quad x_1 \quad x_2 \quad x_1x_2]^T$

$K(x, x) = 1 + x_1^2 + x_2^2 + x_1^2x_2^2$

The above kernel function is a polynomial Kernel function.

5 SVMs and the slack penalty C

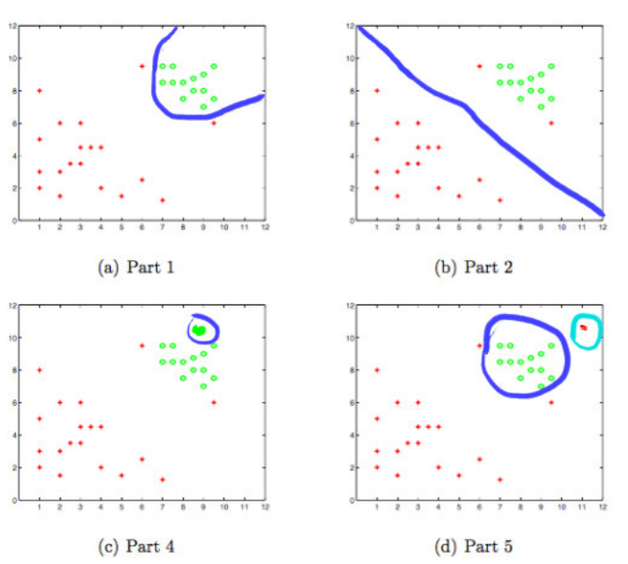

Figure 2: Draw your solutions here.

Figure 2: Solutions to Ques 5

a) The blue boundary in the above figure specifies the one of the possible decison boundaries for large C. When C increases to ∞, we are placing more weights on the slack variables ϵ and hence making a much more stricter classification between the data points. We don't care about the margin in this case and place more weights on the slack of data points with the margin. Therefore, when C $->\infty$, my curve is going to be such that it strictly divides my datapoints into two correctly defined classes, with pure(all correct) classifications in the given sample.

b) The blue boundary in the above figure (b) Part 2, specifies the one of the possible decison boundaries for C approaching to zero. When C goes down to zero, the slack term in our optimization equation comes down to zero, in such a case, we dont care about classification, we rather care about the margin maximisation more, maximum margin between the SVM plane and the support vector is what is aimed for. So, in the given sample, the SVM curve drawn in the figure will divide my dataset properly into two different sets, keeping in mind that more importance is given to margin maximisation than correct classification.

c) In the given dataset, C approaching to zero should work better. Because, it is a more generalised graph and when tested with newer test data set it will get better results. When C approaches to ∞, the curve leads to over-fitting and might not generalise well with new test data points.

d) Adding a green circle on the top right corner as shown in the above figure (c) Part 4, circled in blue, wouldn't lead to change of decision boundaries with large values of C.

e) Adding a red cross at the top right corner, as shown in the above figure, (d)Part 5, will change my curve from a parabolic curve to a circle, classifying the red point correctly.

Adding a green point in the middle section near the red cluster will also change the decision boundary drastically. It will lead to ellipsis kind of a curve instead of a parabolic curve shown earlier.

6 Programming - Bias Variance Trade-off

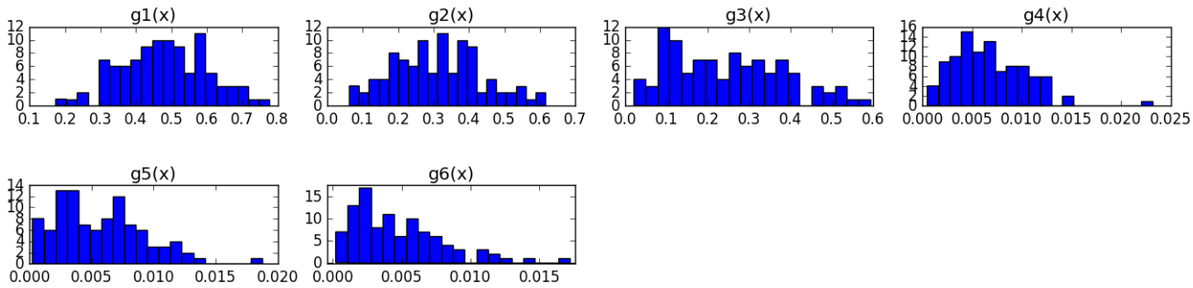

Figure 3: MSE graph for each function, calculated for 100 datasets with 10 datapoints each.

The graph above shows, as the model becomes more complex, the mean square error reduces. As expected. $g_1(x), g_2(x)$ do not depend on X and hence do not fit the data well. The mean square starts reducing down from 2 degree polynomial. i.e $g_4(x) = w_0 + w_1x + w_1x^2$

function	BIAS^2	Variance
g1(x)	0.470301347	0
g2(x)	0.357671956	0.030267498
g3(x)	0.372053677	0.180127798
g4(x)	0.009359101	0.008032482
g5(x)	0.010408258	0.062036413
g6(x)	0.03629889	2.069066871

Figure 4: Bias, Variance w.r.t to different function, 10 datapoints in each dataset

The table above shows, as the model becomes more complex, the bias value reduces. It reduces down at function $g_4(x) = w_0 + w_1x + w_1x^2$, particularly,As the data fits well in this function, the bias increases again a little and den reduces down again as expected- (Bias should decrease with more fitting of data.). As per the variance, it is zero in the beginning as expected, then it starts increasing uptil $g_4(x)$ at which it reduces down again and then starts increasing again.

b)

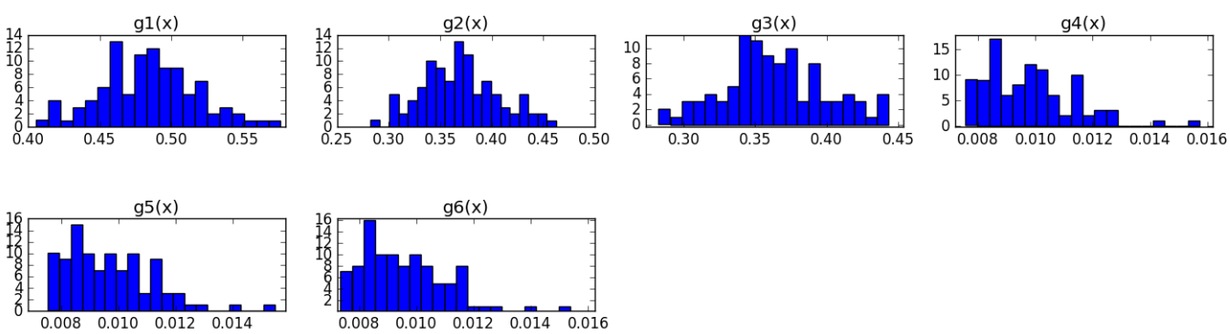

Figure 5: MSE graph for each function, calculated for 100 datasets with 100 datapoints each.

The graph above shows, as the model becomes more complex, the mean square error reduces. The same trend as it was observed with 10 datapoins in each dataset in part a above. As expected. $g_1(x), g_2(x)$ do not depend on X and hence do not fit the data well. The mean square starts reducing down from 2 degree polynomial. i.e $g_4(x) = w_0 + w_1x + w_1x^2$

The table above shows, as the model becomes more complex, the bias value reduces.The same trend as it was observed with 10 datapoins in each dataset in part a above. It reduces down at function $g_4(x) = w_0+w_1x+w_1x^2$,

function	BIAS^2	Variance
g1(x)	0.482432377	0
g2(x)	0.37628659	0.003648342
g3(x)	0.376553133	0.009669851
g4(x)	0.009843056	0.000291785
g5(x)	0.009845204	0.000415072
g6(x)	0.009844829	0.000546796

Figure 6: Bias, Variance w.r.t to different function, 100 datapoints in each dataset

particularly,As the data fits well in this function, the bias increases again a little and den reduces down again as expected- (Bias should decrease with more fitting of data.). As per the variance, it is zero in the beginning as expected, then it starts increasing uptil $g_4(x)$ at which it reduces down again and then starts increasing again.

c)
Given the results shown above, As the number of datapoints increases in the dataset, the $bias^2$ decreases comparatively.Same trend is seen in variance as well. The varaince decreases in 100 points dataset compared to the 10 point datasets. But, the general trend remains the same. Bias, decreases and Variance increase as model complexity increases.

d) The below table shows, that with increase in lambda, as regularization increases, the bias increases. As, Regularizing our function generalises the data and reduces the over-fitting done because of complex models. But, the lambda value more or less remains constant with minor decrease and a little increase at the end. Only minor increase and decreases is observed because the stats given are given for the quadratic function $g_4(x)$, which fits the true data pretty well.

function : w0 + w1*x + w2*x^2		
Lamda	BIAS^2	Variance
0.001	0.008913027	0.000318069
0.003	0.008912461	0.000317977
0.01	0.008910791	0.000317669
0.03	0.008908653	0.000316906
0.1	0.008931235	0.000315565
0.3	0.00923651	0.000322006
1	0.01255532	0.000430231

Figure 7: Bias, Variance w.r.t to g4(x) function, for different lambda values

7 Linear and Kernel SVM Programming

a)

C	CV Accuracy	Avg Time
0.00024414	55.75	0.322884
0.00097656	55.75	0.330652
0.00390625	55.75	0.31948169
0.015625	71.45	0.31346965
0.0625	91.65	0.226415
0.25	91.75	0.14234829
1	94.25	0.09751264
4	94.9	0.07352726
16	96.15	0.07204962

Figure 8: 3-Fold Cross validation accuracy and average training time

The above table shows, as C increases i.e. slack variable weight increases, miss-classification is allowed and over-fitting reduces down, Accuracy increases. Average time also decreases as C increases.The maximum accuracy 96.15% is observed at C = 16

b)

C	Degree	Avg Time	CV Accuracy
0.015625	1	0.33372768	55.75
0.015625	2	0.32183568	55.75
0.015625	3	0.32308133	55.75
0.0625	1	0.2506543	89.9
0.0625	2	0.30660129	88.4
0.0625	3	0.33825095	72.2
0.25	1	0.1713376	91.15
0.25	2	0.20662173	92
0.25	3	0.25695999	91.6
1	1	0.11552461	93.2
1	2	0.13728166	92.75
1	3	0.15529466	92.8
4	1	0.08258533	94.2
4	2	0.08855001	95.05
4	3	0.10906633	94.95
16	1	0.07815496	94.5
16	2	0.07047733	96.2
16	3	0.08089701	96.25
64	1	0.07500696	94.55
64	2	0.07052604	96.35
64	3	0.06856934	96.4
256	1	0.09947666	94.35
256	2	0.07020402	96.4
256	3	0.07446941	96.4
1024	1	0.22683271	94.9
1024	2	0.07606999	96.3
1024	3	0.07041693	96.2
4096	1	0.6410013	93.95
4096	2	0.08363668	95.95
4096	3	0.07764133	95.55
16384	1	1.95357267	94.4
16384	2	0.08394829	96.55
16384	3	0.06811237	96.6

Figure 9: 3-Fold Cross validation accuracy and average training time for Polynomial Kernel

Maximum Accuracy for Polynomial Kernel SVM Regression is observed at C = 16384, Degree =3, Accuracy = 96.6 %, In general higher Accuracy is observed at higher C values and higher degree.

C	gamma	Avg Time	CV Accuracy
0.015625	6.10E-05	0.33669766	55.75
0.015625	0.00024414	0.31966964	55.75
0.015625	0.00097656	0.31306005	55.75
0.015625	0.00390625	0.31632702	55.75
0.015625	0.015625	0.31630039	55.95
0.015625	0.0625	0.31549803	87.6
0.015625	0.25	0.31896265	60.25
0.0625	6.10E-05	0.32708136	55.75
0.0625	0.00024414	0.31727099	55.75
0.0625	0.00097656	0.3139046	55.75
0.0625	0.00390625	0.32176105	64.4
0.0625	0.015625	0.25127935	90.5
0.0625	0.0625	0.20355439	91.85
0.0625	0.25	0.27854872	92.5
0.25	6.10E-05	0.32437865	55.75
0.25	0.00024414	0.34493637	55.75
0.25	0.00097656	0.33974671	66.65
0.25	0.00390625	0.26313233	90.7
0.25	0.015625	0.160906	91.2
0.25	0.0625	0.12855562	93.6
0.25	0.25	0.19810502	95.6
1	6.10E-05	0.34357198	55.75
1	0.00024414	0.32540472	67.45
1	0.00097656	0.24036098	90.6
1	0.00390625	0.15122334	91.15
1	0.015625	0.10817305	93.55
1	0.0625	0.09503198	95.75
1	0.25	0.12167637	97.75
4	6.10E-05	0.32906493	67.65
4	0.00024414	0.25380802	90.3
4	0.00097656	0.15047661	91.3
4	0.00390625	0.10947967	93.7
4	0.00390625	0.10947967	93.7
4	0.015625	0.08471266	95.05
4	0.0625	0.07180802	96.5
4	0.25	0.11726801	97.6
16	6.10E-05	0.24589594	90.7
16	0.00024414	0.15441537	91.55
16	0.00097656	0.10902834	93.35
16	0.00390625	0.07830501	94.45
16	0.015625	0.0700984	95.65
16	0.0625	0.07128533	96.85
16	0.25	0.11463102	96.8
64	6.10E-05	0.15605632	91.25
64	0.00024414	0.11554734	93.45
64	0.00097656	0.08473468	94.3
64	0.00390625	0.07228764	94.8
64	0.015625	0.06982764	95.95
64	0.0625	0.06765032	96.85
64	0.25	0.11785865	96.8
256	6.10E-05	0.10488701	93.75
256	0.00024414	0.08754603	94.25
256	0.00097656	0.07761733	94.8
256	0.00390625	0.07480661	95.05
256	0.015625	0.07620366	97.2
256	0.0625	0.06878066	96.8
256	0.25	0.11894735	96.8
1024	6.10E-05	0.09108226	94.45
1024	0.00024414	0.07625302	94.5
1024	0.00097656	0.07650669	94.65
1024	0.00390625	0.08612434	96.1
1024	0.015625	0.07963371	96.75
1024	0.0625	0.0725867	96.55
1024	0.25	0.11776463	96.55
4096	6.10E-05	0.07268063	94.4
4096	0.00024414	0.07773733	94.85
4096	0.00097656	0.09467538	95.4
4096	0.00390625	0.11191233	96.65
4096	0.015625	0.07850369	96.8
4096	0.0625	0.0712467	96.15
4096	0.25	0.11285162	96.3
16384	6.10E-05	0.07727933	94.25
16384	0.00024414	0.10601465	95.15
16384	0.00097656	0.15150436	96.5
16384	0.00390625	0.14219666	96.8
16384	0.015625	0.08771229	97.25
16384	0.0625	0.06958469	96.65
16384	0.25	0.11351705	96.65

Figure 10: 3-Fold Cross validation accuracy and average training time for RBF Kernel

c) Maximum Accuracy for RBF Kernel SVM Regression is observed at C = 4, gamma =0.25, Accuracy = 97.75. The same parameters have been configured in libsvm.py as well, which will test the accuracy on test

data using these values for the RBF Kernel on top of SVM

d) When I test the data using the best parameters found after training on RBF as well as Polynomial kernel, RBF Kernel with parameters C = 4, gamma =0.25 gives the best results. After configuring the SVM model with RBF Kernel and above paramters i get Accuracy = 95.55% (1911/2000) (classification)