Support Vector Machines

Nikhil Jangamreddy (2018csm1011)

March 29, 2019

1 Implementation of SVM Using CVOXPT package :

1.1 Problem Statement

In this experiment, we implement Support Vector Machine using CVOXPT package. We then implement Linear Kernel, Polynomial Kernel and Gaussian Kernel. We Create random dataset based on Linearly Separable data, Linear separable Overlapping data, and Circular Data. Later we split the data into Train and Test data based on Fraction 0.75. Later we call Linear SVM, Kernel SVM and Soft SVM based on Different Kernel Function and Kernel Hyper Parameters.

2 Implementing Quadratic Problem Solver :

In Implementation, we have used CVOXPT package to Implement SVM optimisation called SMO. The CVOXPT solver takes parameters - q, P, G, h, A, and b as Inputs to find the solution. Using the solution we define the Fit function. We generate Gram Matrix to Find the Fit in the SVM formulation. In this we use Lagrange Multipliers to pose Optimisation problem.
To obtain the Lagrange Multiplier solution in CVOXPT we use : alpha = np.ravel(solution['x']) code.

3 Generating Linear Separable, Linear Separable Overlapping data and Circular Data :

In Implementation, we have used Random generation of points. For linearly Separable data, we have generated 100 Positive and 100 Negative points that can be linearly Separated.
For Linear Separate Overlap data, we create data that is almost linearly separated i.e.., 100 instances of Positive and 100 instances of Negative points. For Circular data, we generate 100 Instances of Positive points and 100 instances of Negative points that can be separated by Circular decision Boundary.

4 Generating Linear Separable, Linear Separable Overlapping data and Circular Data :

In Implementation, we have used Random generation of points. For linearly Separable data, we have generated 100 Positive and 100 Negative points that can be linearly Separated.
For Linear Separate Overlap data, we create data that is almost linearly separated i.e.., 100 instances of Positive and 100 instances of Negative points. For Circular data, we generate 100 Instances of Positive points and 100 instances of Negative points that can be separated by Circular decision Boundary.

5 Implementing Different Kernel Functions :

In Implementation, we have used Different Kernel Functions : Linear Kernel, Polynomial Kernel and Gaussian Kernel. For each Kernel Function Depending on the Hyper parameters, we get different Decision Boundaries. By using Kernel Functions, we are transforming the data into higher Dimension where Data is Linearly Separable.
For Polynomial Kernel the power q is the Hyper parameter which we can vary to Vary the Plot that fits the data. Similarly for Gaussian Kernel, we have width s as Hyper parameter.

6 Implementing Different Kernel Functions :

In Implementation, we have used Different Kernel Functions : Linear Kernel, Polynomial Kernel and Gaussian Kernel. For each Kernel Function Depending on the Hyper parameters, we get different Decision Boundaries. By using Kernel Functions, we are transforming the data into higher Dimension where Data is Linearly Separable.
For Polynomial Kernel the power q is the Hyper parameter which we can vary to Vary the Plot that fits the data. Similarly for Gaussian Kernel, we have width s as Hyper parameter.

7 Implementing Training Phase in SVM :

In Implementing Training Phase in SVM, we Compute Gram matrix and Necessary parameters q, P, G, h, A, and b are passed to Convex Optimiser in CVOXPT package through the line : solution = cvxopt.solvers.qp(P, q, G, h, A, b). Since SVM is Convex Optimisation problem, we are Guaranteed to reach Global Minima.
For Calculating Lagrange Multipliers i.e.., Alpha values in SVM formulation we use alpha = np.ravel(solution['x']). We constrain the Number of Support vectors if the Alpha value is very less.

8 Implementing Predict Function for Class Labels :

In Implementing Predict Function we calculate the $W^t x + w_0$ value and determine the sign and Based on the sign we Assign the class Label to it. Given x and W Optimised based on SMO To predict Class Label:
If $W^t x + w_0 >= 0$ then Positive Class.
Else $W^t x + w_0 < 0$ then Negative Class.

9 Implementing Linear SVM, Kernel SVM and Soft SVM Functions :

In Implementing linear SVM, we have taken different Data sets i.e.,, Linearly Separable, Linearly Separable Overlap data and Circular data. Implementation Involves Splitting the data into Train and Test data using Train test Fraction = 0.75 Later building Model based on the Kernel Chosen and Finding the Accuracy on the Model Built.

10 Plots for Different Kernels w.r.t to Different Data sets :

For **Linearly Separable Data :**

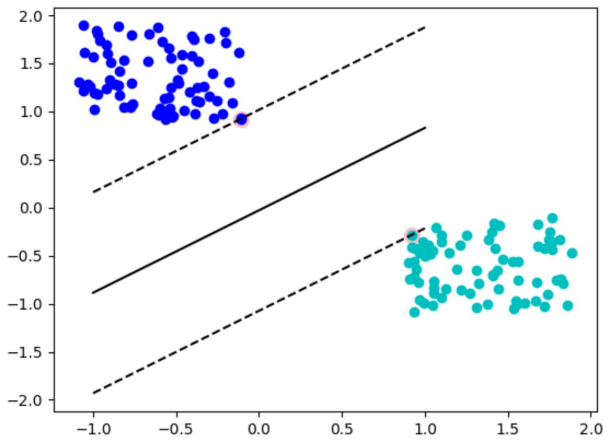

Fig 1 : Linear Kernel

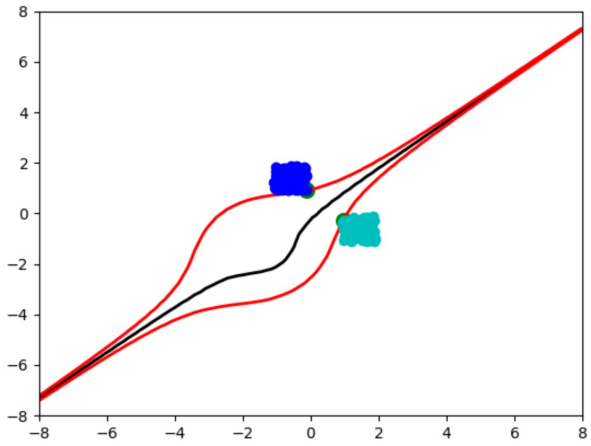

Fig 2 : Polynomial Kernel

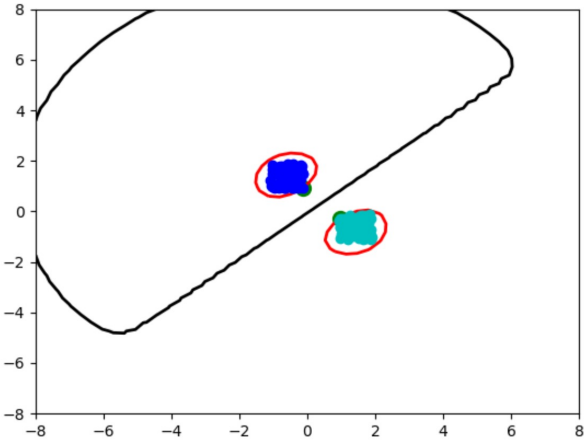

Fig 3 : Gaussian Kernel

10.1 Observations :

- In case of Linearly Separable data, Gaussian Kernel led to a highly Over fitted model.
- In general Linear Kernel performed better generalisation performance.

For **Linearly Separable Overlap Data :**

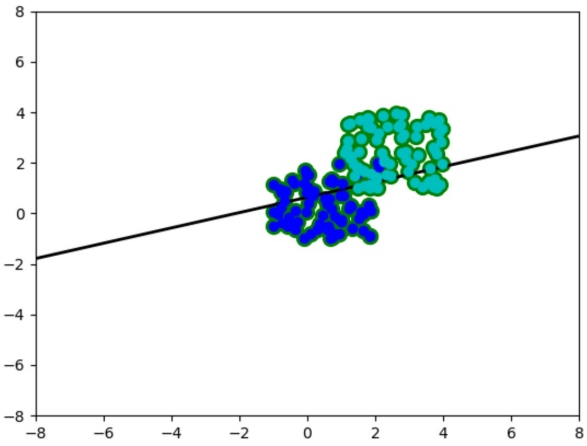

Fig 4 : Linear Kernel

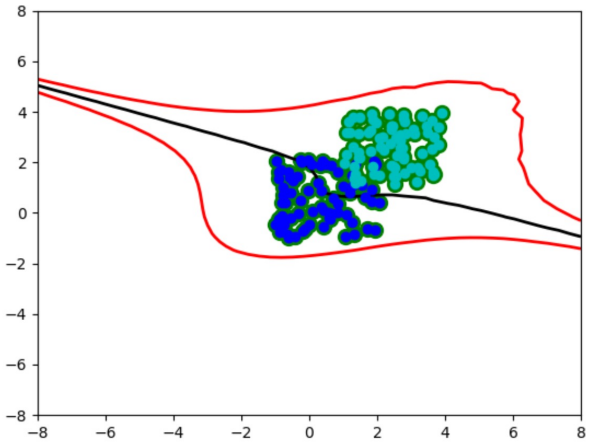

Fig 5 : Polynomial Kernel

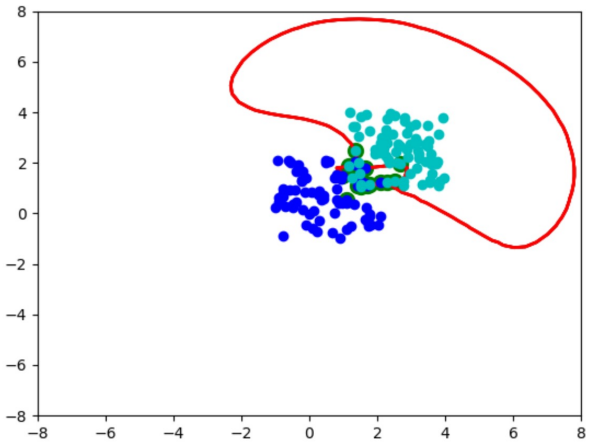

Fig 6 : Gaussian Kernel

10.2 Observations :

- In Linear Separable Overlap data, Linear kernel gave a very High Bias model.
- Polynomial Kernel and Gaussian Kernel gave better decision Boundaries compared to Linear Kernel.

For **Circular Data :**

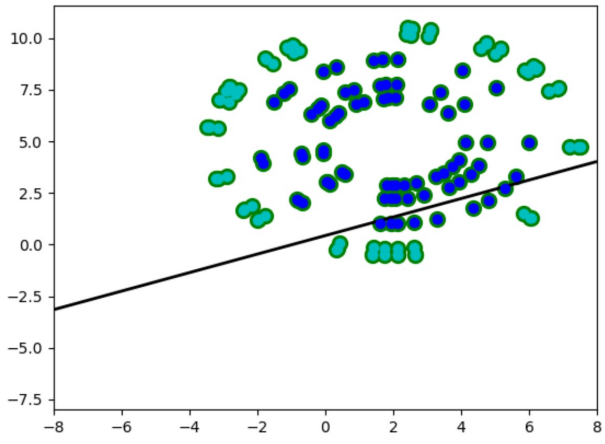

Fig 7 : Linear Kernel

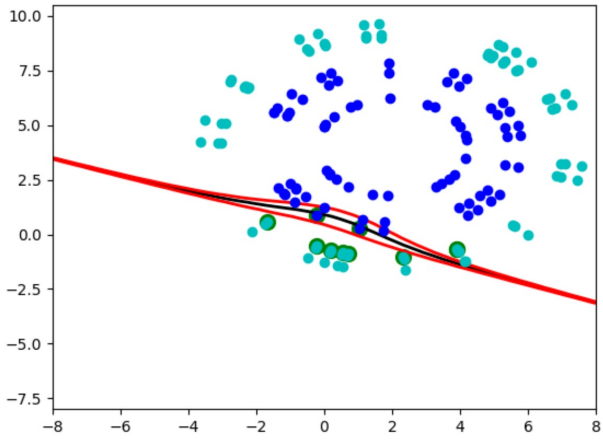

Fig 8 : Polynomial Kernel

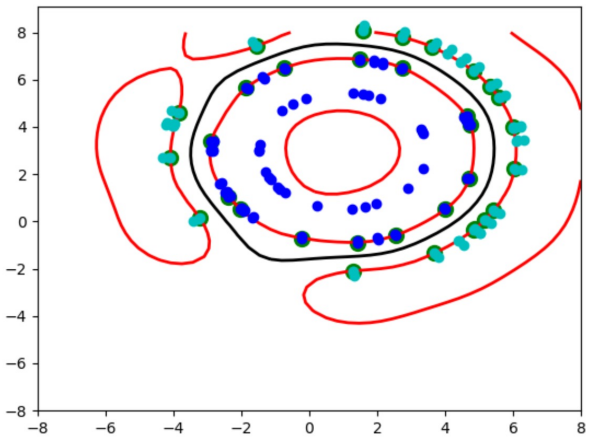

Fig 9 : Gaussian Kernel

10.3 Observations :

- In Circular data, Linear Kernel and Polynomial Kernel gave high Bias model.
- Gaussian Kernel gave better Decision Boundary Compared to Linear and Polynomial Kernel data.

11 Implementing Linear SVM, Kernel SVM and Soft SVM Functions :

In Implementing linear SVM, we have taken different Data sets i.e.,, Linearly Separable, Linearly Separable Overlap data and Circular data. Implementation Involves Splitting the data into Train and Test data using Train test Fraction = 0.75 Later building Model based on the Kernel Chosen and Finding the Accuracy on the Model Built.

12 Plots By varying Hyper Parameters :

For **Linearly Separable Overlap Data :**
Polynomial Kernel : $(1 + x1.x2)^q$ **Hyper Parameter : q**

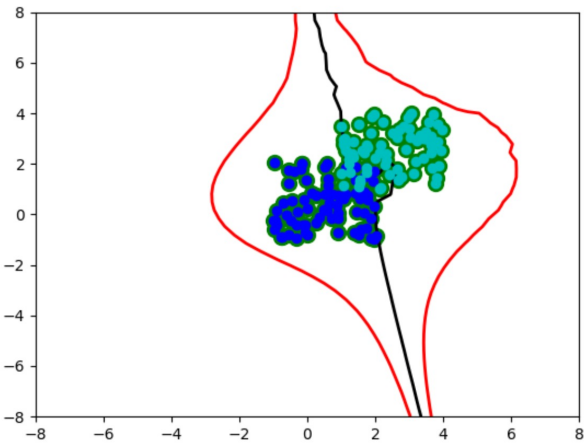

Fig 10 : Polynomial Kernel q = 3

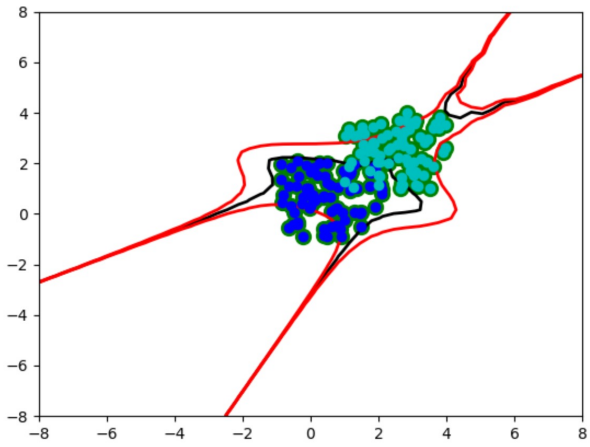

Fig 11 : Polynomial Kernel q = 4

12.1 Observations :

- In Linearly Separable Overlapping data, as q increases the model Becomes overfit.
- The choice of q is Decided based on Trade off Between Variance and Bias.

For **Linearly Separable Data :**
Gaussian Kernel : $e^(- ((x1 - x2)_2)^2/2 * s^2)$ **Hyper Parameter : s**

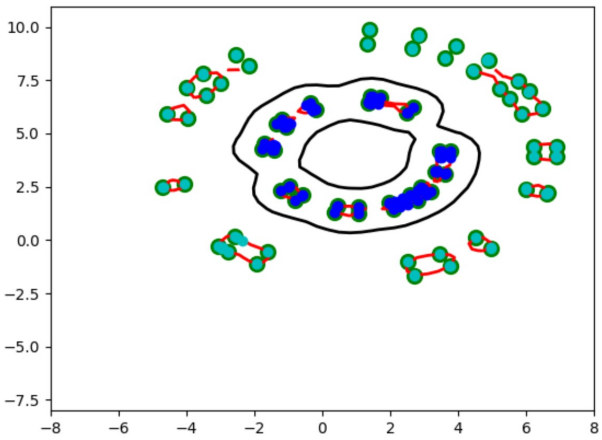

Fig 12 : Gaussian Kernel s = 0.8

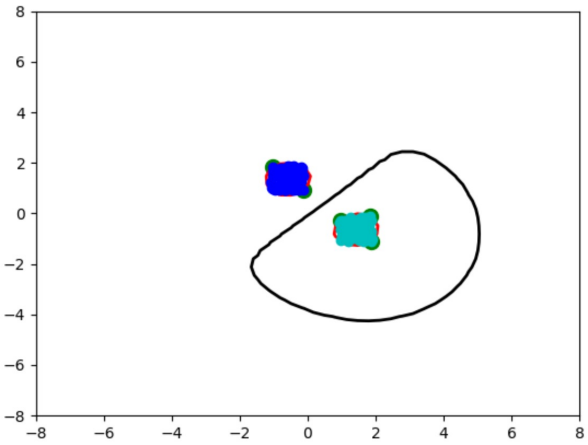

Fig 13 : Gaussian Kernel s = 1.5

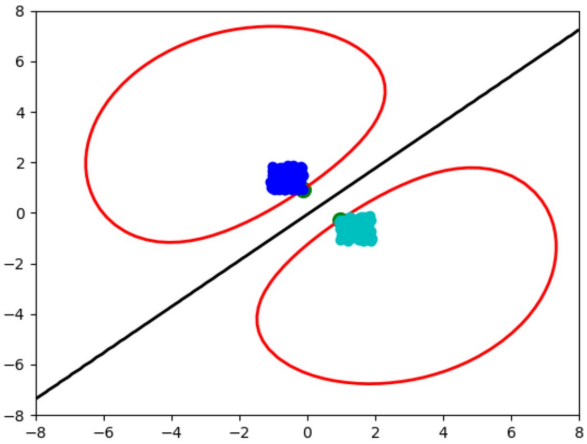

Fig 14 : Gaussian Kernel s = 2

12.2 Observations :

- As s Decreases, model is becoming highly Overfit.
- Fix a high s value, slowly decrease it based on Validation data performance and based on bias performance.
- As s decreases even more, each point becomes a support vector.

13 Implementing SVM On IRIS Data set:

In order to Implement SVM on Iris data, we have to Keep In mind this is not 2 class classification rather it is 3 class Classification. So we adopt **One Vs All approach**. For each case out of 3 cases :
For each Kernel Function : 1) Linear kernel, 2) Polynomial kernel and 3) Gaussian Kernel
Case 1 : Class 1 = 1 Remaining Classes = -1
Case 2 : Class 2 = 1 Remaining Classes = -1
Case 3 : Class 2 = 1 Remaining Classes = -1
out of 3 predictions :
we take argmax$g_k(x)$ as Class prediction.
Where $g_k(x)$ Is prediction based on weights of SVM for each case in Above cases.
For **Linear Kernel :** Predictions for 3 cases :
For Case 1 :
[-2.6424666 1.32492541 -5.53802684 -2.52431198 -2.8678435 1.28426169 -1.51021401
-3.43464159 -2.89872484 -1.87829685 -3.22480134 1.34276404 1.62529937 1.29001637
1.55317924 -2.57648428 -4.12419699 -1.92701639 -2.46983776 -4.02330515 1.20466117
-3.017263 1.20235921 -3.97688714 -4.28033037 -3.57792332 -4.20859328 -4.18039738

1.24992802 1.14788541 2.01985239 1.79217389 -2.30545743 1.30440221 1.51942097
-3.43406578 -2.38620914 1.43847696]
For Case 2 :
[-0.30367929 -0.30374132 -0.30379429 -0.30357898 -0.3038932 -0.3038443 -0.30362702
-0.30357767 -0.30399907 -0.30376252 -0.30342676 -0.30388175 -0.30390357 -0.30385206
-0.30355044 -0.30344254 -0.30336212 -0.30380161 -0.30354765 -0.30345947 -0.30369171
-0.30344362 -0.30367388 -0.3034762 -0.30342299 -0.30353237 -0.30375079 -0.30334251
-0.3038483 -0.30377886 -0.30356809 -0.30346662 -0.30378174 -0.30363137 -0.30363328
-0.30373016 -0.30358341 -0.30375263]
For Case 3 :
[-0.32227566 -0.3221611 -0.3197472 -0.32156416 -0.32111925 -0.32227214 -0.32179582
-0.31918451 -0.32156464 -0.32206741 -0.32025492 -0.32366082 -0.32241126 -0.32358707
-0.32269925 -0.32102169 -0.32039872 -0.32257634 -0.32237015 -0.32043226 -0.32363952
-0.32106181 -0.3228009 -0.32064798 -0.31941053 -0.31951795 -0.32119679 -0.31984866
-0.3232294 -0.32355818 -0.32318585 -0.32160388 -0.32084392 -0.32346028 -0.32376289
-0.32085417 -0.32101035 -0.32289889]

Number of **Correct prediction by multi class SVM using Kernel function : Linear Kernel is : 68.42.**
For **Polynomial Kernel :** Predictions for 3 cases :
For Case 1 :
[-4.53653505 1.37942007 -14.7148268 -4.26013009 -5.79273718 1.27950066 -2.15737852
-7.54537939 -4.63145572 -2.79081059 -6.60200324 1.26675275 1.62384032 1.2374661
1.54792666 -4.99363879 -8.63293144 -2.6015616 -3.79856482 -7.96333368 1.16809526
-5.44893714 1.18608016 -7.85616693 -12.75899035 -7.51472179 -8.5809838 -9.53925983
1.19572895 1.12006824 1.80591282 1.99042592 -4.67161169 1.26529602 1.38190809
-6.11716047 -4.61414749 1.41857996]
For Case 2 :
[-0.90206912 -27.06912113 -76.83928374 8.45040003 2.56320159 -4.09140156 17.57321759
-25.46718736 10.941444 13.20851948 -10.78429177 -31.12779606 -23.26713933 -34.18910317
-31.20134038 7.44279354 -33.84884767 8.9299365 7.73432859 -35.09099303 -29.67315093
-1.51099816 -13.69225786 -26.51845726 -46.02368665 -33.64081036 -19.01269462
-43.50387615 -13.07314388 -26.70796289 -37.32295799 -33.74429159 14.52309856
-33.49780789 -29.95874452 -8.51069305 10.77575962 -31.84309652]
For Case 3 :
[-0.27400643 -67.8771743 63.14942489 -9.57489294 -3.94512651 -72.35332553 -24.67386096
27.26508741 -10.79410779 -17.68634677 11.56363265 -27.92935651 -72.43000188
-27.14327787 -56.87245229 -9.03509685 30.86576476 -11.91653188 -8.51458701 31.92327543
-26.21798511 1.65810134 -50.88044931 23.93177294 48.59827868 34.12254123 16.28342128
41.58603979 -42.43238465 -27.91685375 -51.76107819 -97.67865815 -19.53194744
-30.97758908 -29.04451341 7.93033201 -13.96750923 -48.61512059]

Number of **Correct prediction by multi class SVM using Kernel function : Polynomial Kernel is : 97.36.**

For **Gaussian Kernel :** Predictions for 3 cases :
For Case 1 :
[-1.16632139 0.96493473 -0.930146 -1.22188599 -1.06562447 1.10347573 -1.24918956
-1.0337838 -1.17061024 -1.31414098 -1.08994772 1.24181825 1.17080173 1.21836236
1.21663957 -1.10304722 -1.07298581 -1.35339378 -1.2491873 -1.05453189 1.18115341
-1.13824462 1.15361533 -1.06438777 -0.98647662 -1.05564716 -1.02740549 -1.09358692
1.21082358 1.16022039 1.25696318 0.90423498 -1.06085261 1.21233097 1.24143276
-1.09411446 -1.11235905 1.22017404]
For Case 2 :
[7.42397577 -1.45714164 -9.43594858 6.13930685 7.08428085 -2.29032576 9.99796678
1.05199739 -0.3223068 10.43883212 -5.65381146 -2.797402 -3.64737281 -2.52833566
-3.81827966 4.09010321 -14.30056782 9.36555113 11.52173612 -10.75792822 -2.44798581
-1.72058277 -2.36191767 -9.98975796 -9.29406123 -5.29951415 -7.66045072 -15.51659765
-2.59095396 -1.95706003 -5.46505134 -2.98000683 12.92930685 -2.94185959 -4.03886357
-3.72965662 8.07778771 -3.19906448]
For Case 3 :
[-7.34871638 -6.95137284 9.09927131 -6.07884946 -7.05931983 -6.38475446 -11.30030434
-0.88350843 0.47578979 -10.97637775 5.89988781 -5.44898825 -4.83190138 -6.03852289
-5.43789461 -4.00856674 14.34391871 -9.61081881 -11.41112542 10.8074891 -6.28442697
1.94334033 -6.70576999 10.07061599 8.9547126 5.47421753 7.73718511 15.62133069
-5.62135548 -6.60179684 -2.85244729 -4.60402769 -13.24374649 -6.1995828 -4.16624822
3.87847531 -8.18846341 -5.90720452]

Number of **Correct prediction by multi class SVM using Kernel function : Gaussian Kernel is : 94.73.**