E9-253: Neural Networks and Learning Systems-I
HW-4
Submitted by - Ocima Kamboj
Serial Number - 06-02-01-10-51-18-1-15899

Problem-6.25 Support Vector Machines

Methodology

Experimental Setup

The program for the Support Vector Machine was done is MATLAB. We constructed a Quadratic function to minimize following the dual problem (as per Haykin)-

$$\max Q(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i=1}^{N} \alpha_i \alpha_j d_i d_j \Phi^T(\boldsymbol{x_i})\Phi(\boldsymbol{x_j})$$

subject to the constraints -

$$(1)\ \sum_{i=1}^{N} \alpha_i d_i = 0$$
$$(2)\ \alpha_i \geq 0\ for\ i = 1, 2, ...N$$

The optimization framework is implemented as follows -
Let $\{\boldsymbol{x_i}, d_i\}_{i=1}^N$ be the training set. Let D denote a diagonal matrix with $D_{ii} = d_i$. Let $\boldsymbol{K}$ be the Gram Matrix. let e be a vector with all the elements equal to one. Then, the objective function is -

$$Q(\boldsymbol{\alpha}) = e^T\boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T D^T \boldsymbol{K} D\boldsymbol{\alpha}$$

subject to the constraints -

$$(1)\ \sum_{i=1}^{N} \alpha_i d_i = 0$$
$$(2)\ \alpha_i \geq 0\ for\ i = 1, 2, ...N$$

where $\boldsymbol{\alpha}$ is the vector of $\alpha_i's$.
We use the 'quadprog' function of MATLAB to solve this quadratic optimization problem.
'quadprog' gives us a vector of $\alpha_i's$. The $\alpha_i's$ that are greater than 0 correspond to those inputs that are in the support. The $\alpha_i's$ for which $0 \leq \alpha_i \leq C$ correspond to those inputs which belong to the unbounded support. As there are some numerical inaccuracies, we set a small tolerance ϵ to judge the values of α. We make all $\epsilon \leq \alpha_i$ equal to 0. The value of the bias is calculated as follows -

$$b(s) = d(s) - \sum_{i=1}^{N_s} \alpha_{0,i} d_i \Phi^T(\boldsymbol{x_i})\Phi(\boldsymbol{x}(s))$$

or equivalently

$$b(s) = d(s) - \sum_{i=1}^{N_s} \alpha_{0,i} d_i \boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{x}(s)$$

Here, s refers to those $\{\boldsymbol{x_i}, d_i\}_{i=1}^{N}$ from the training set which belong to the unbounded support. Thus $d(s)$ is either equal to 1, or -1. The summation is over all the support vectors. N_s is the number of support vectors. $\boldsymbol{K}$ is the kernel.$\alpha_{0,i}$ correspond to the optimum value of α.
Then we get our bias as -

$$b = \frac{\sum_{i=1}^{N_s} b(s)}{N_s}$$

The predicted outcome y_n for some x_n is calculated as -

$$y_n = \sum_{i=1}^{N_s} \alpha_{0,i} d_i \Phi^T(\boldsymbol{x_i})\Phi(\boldsymbol{x_n}) + b$$

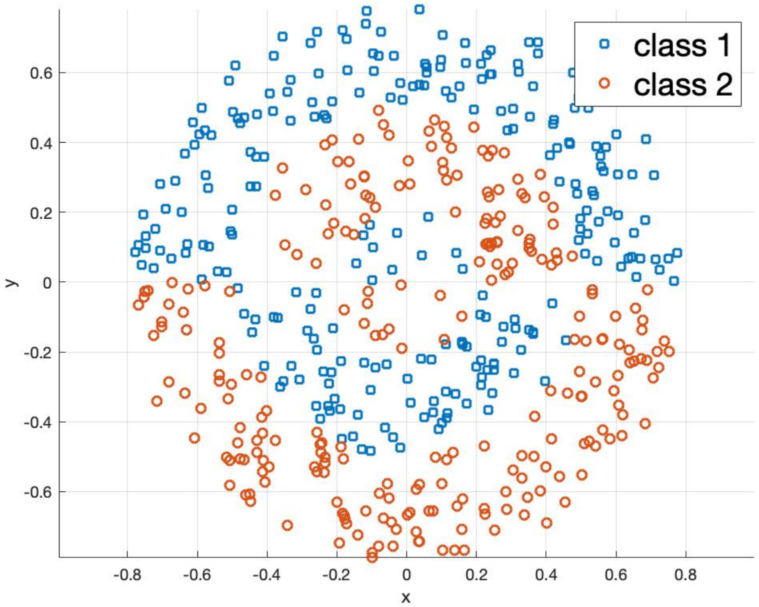
or

$$y_n = \sum_{i=1}^{N_s} \alpha_{0,i} d_i \boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{x_n}) + b$$

Generation of Data Points
$N = 500$ data points were generated as specified in the question, with 250 points in each class. $d_1 = 0.2$, $d_2 = 0.5$, and $d_3 = 0.8$.

Generation of Testing Data Points
$N = 1000$ data points were generated as with the same distribution as the training examples, with 500 points in each class. $d_1 = 0.2$, $d_2 = 0.5$, and $d_3 = 0.8$.

(a) Aim Generate the Data.

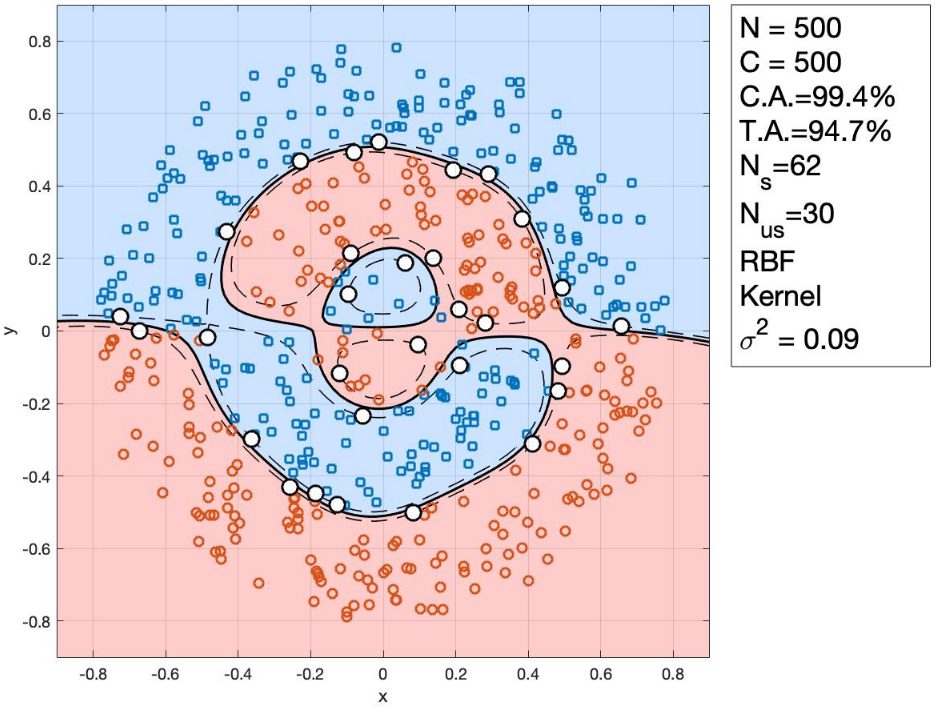


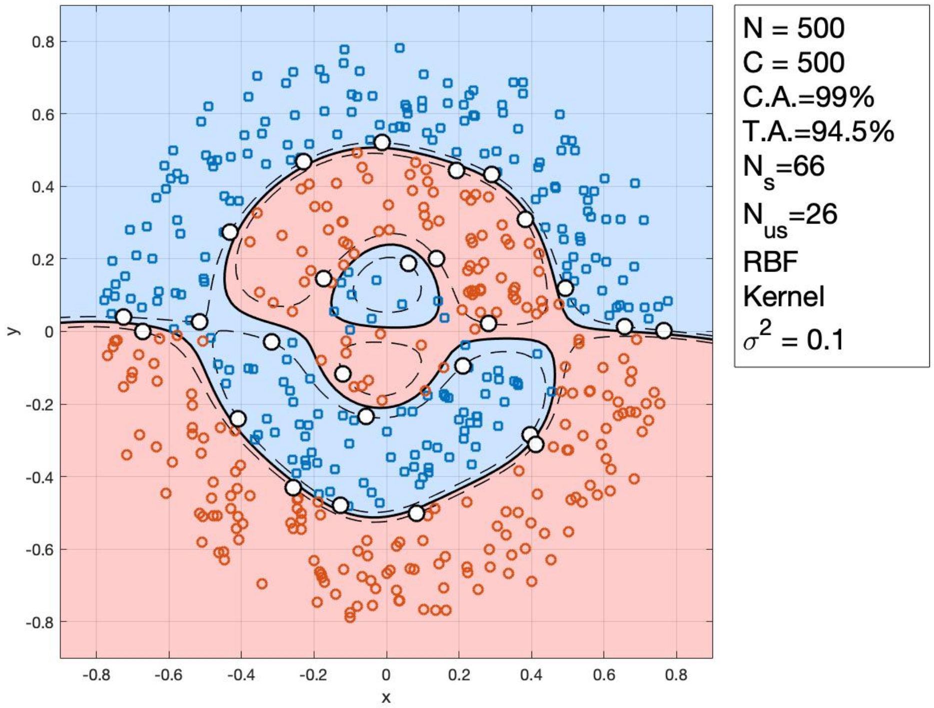
(b) Aim Train a support vector machine,assigning the value C=500

We experiment with the RBF kernel and the polynomial kernel. The results have been plotted below. In all the figures, the terminology is as follows - N is the number of training examples, C is the parameter that puts a constraint on the possible values of α, C.A. implies the classification accuracy (as achieved on the training set), T.A is the testing accuracy, N_s is the number of exemplars in the support, N_{us} is the number of elements belonging to the unbounded support. The dotted line corresponds to the margin of separation. The white coloured points are the *unbounded* support vectors.

1. RBF Kernel

$$\boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{x_j}) = exp\Big(- \frac{1}{2\sigma^2} \|\boldsymbol{x_i} - \boldsymbol{x_j}\|^2 \Big)$$

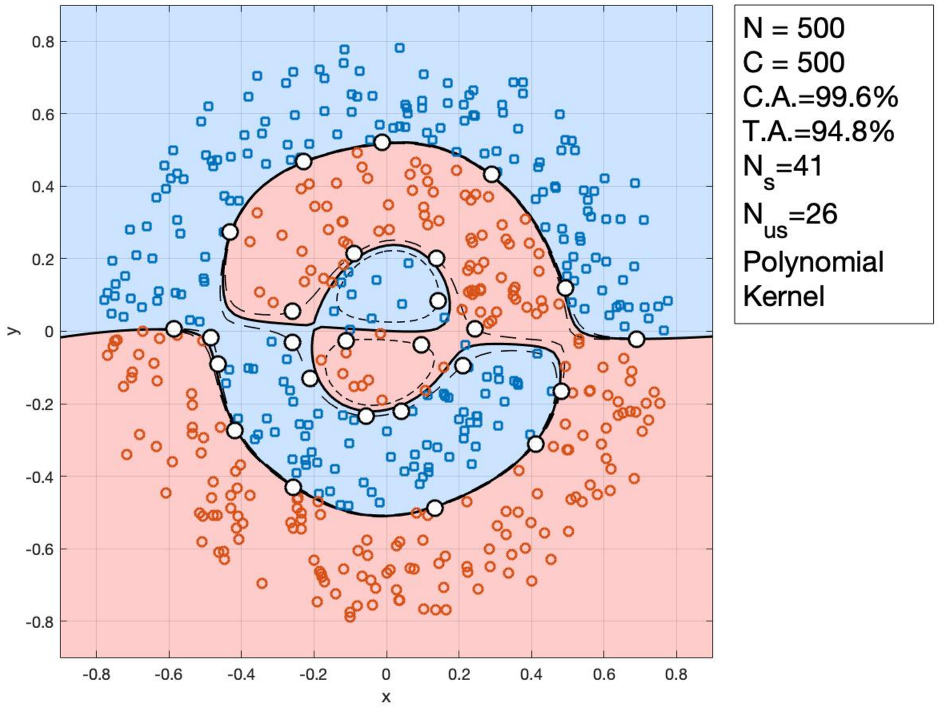




1. Polynomial Kernel

$$\boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\boldsymbol{x_i}^T \boldsymbol{x_j} + 1)^p$$

We chose $p = 16$ because it was giving good results.



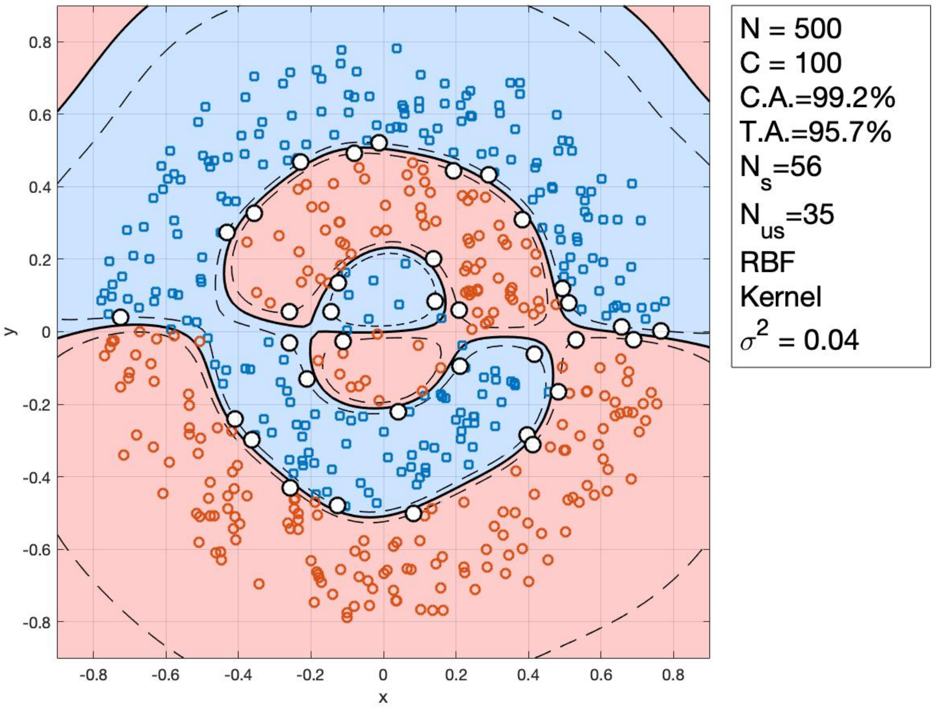
Observations - At the end.

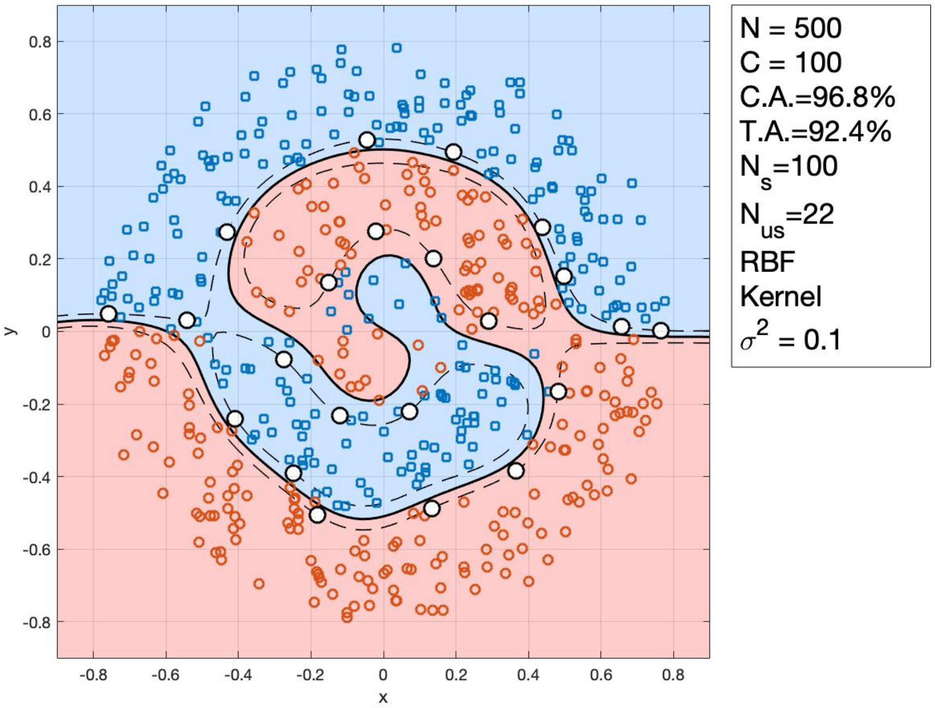
(c) Aim Test the network and thereby determine the classification error rate

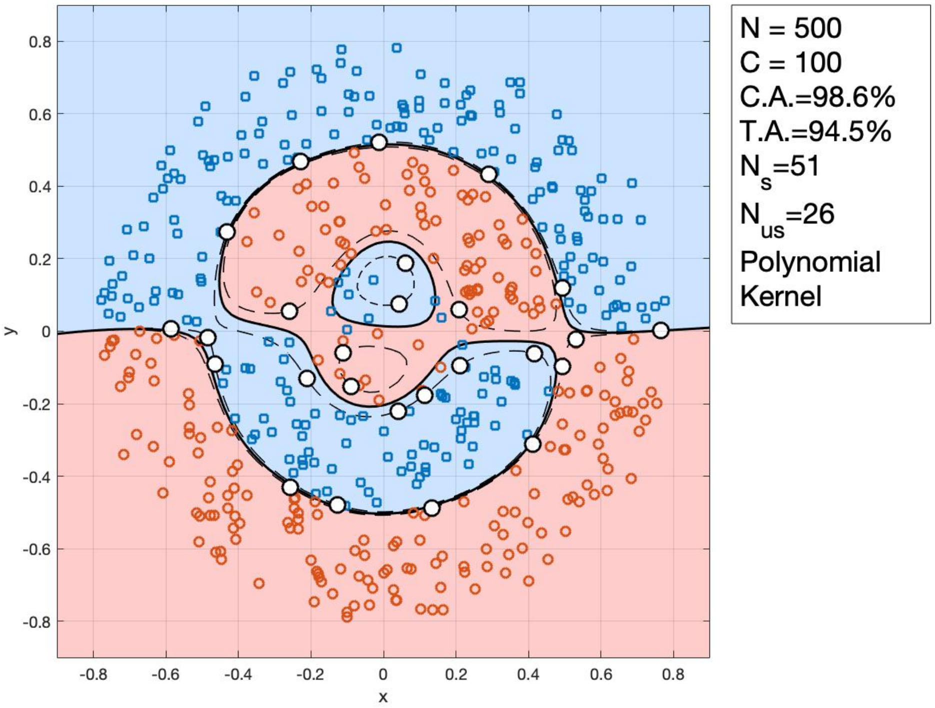
We test the network with 1000 data points. As mentioned in the previous question, the classification accuracy on the test set for the case of RBF kernel is 94.7% and for the case of polynomial kernel is 94.8%.

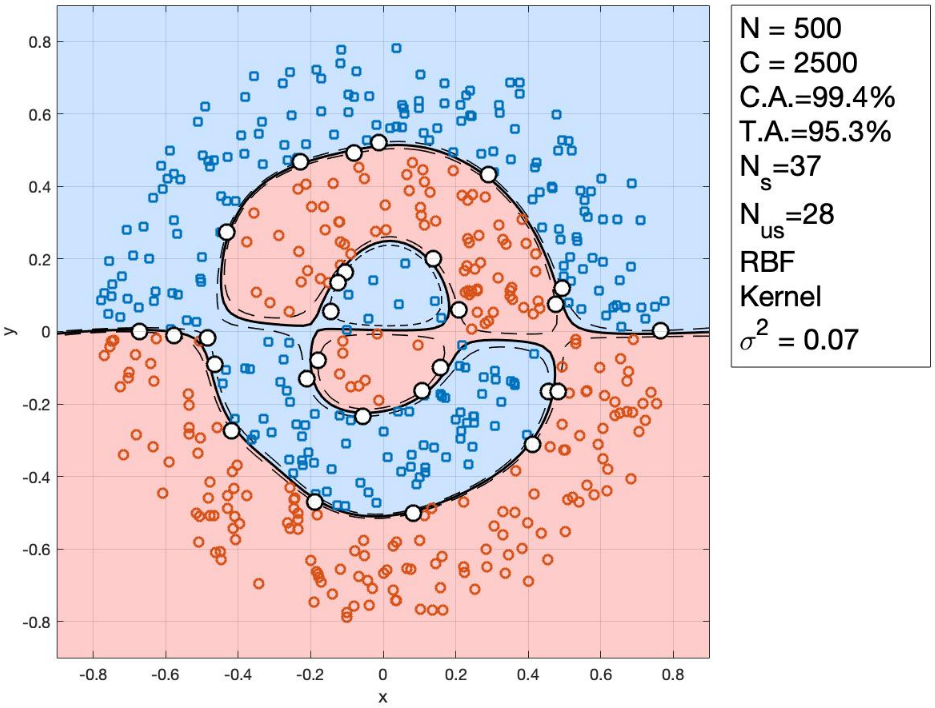
Thus the classification error rate for RBF kernel is 5.3%, and for polynomial kernel is 5.2%.

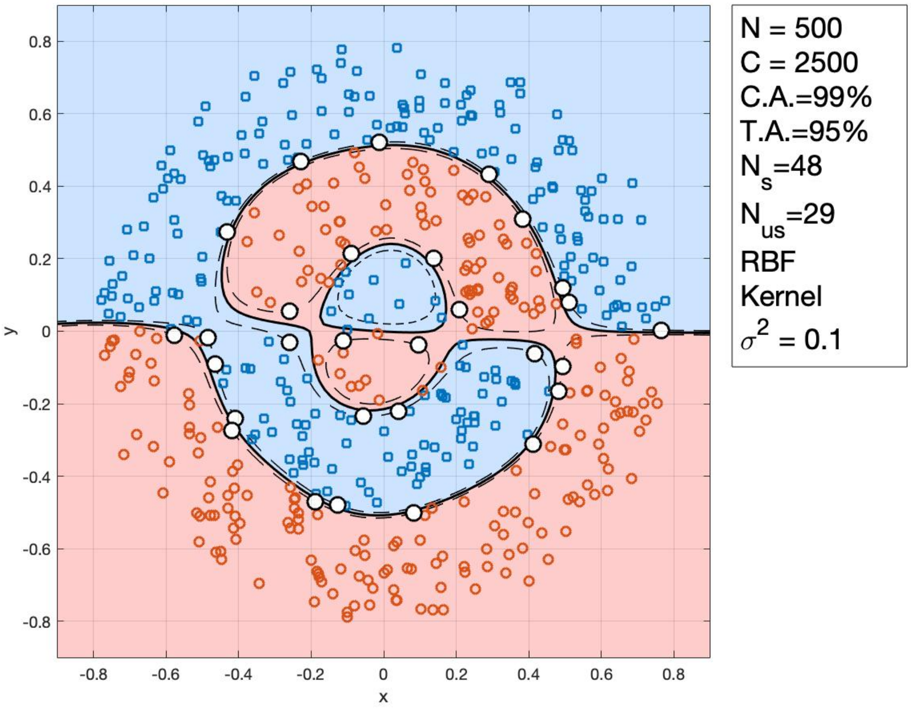
(d) Aim Repeat the experiment for C = 100 and C = 2,500.
The results have been plotted below.

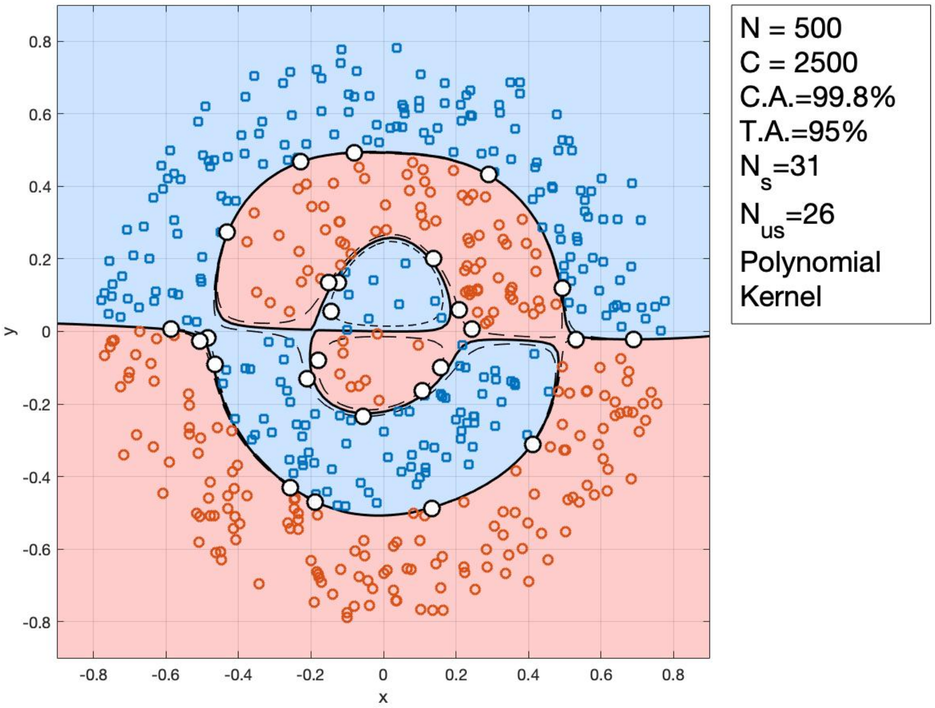



N = 500
C = 100
C.A.=96.8%
T.A.=92.4%
N_s=100
N_{us}=22
RBF
Kernel
σ^2 = 0.1
y
x
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8


N = 500
C = 100
C.A.=98.6%
T.A.=94.5%
N_s=51
N_{us}=26
Polynomial
Kernel
y
x
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8


N = 500
C = 2500
C.A.=99.4%
T.A.=95.3%
N_s=37
N_{us}=28
RBF
Kernel
σ^2 = 0.07
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
y
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
x


N = 500
C = 2500
C.A.=99%
T.A.=95%
N_s=48
N_{us}=29
RBF
Kernel
σ^2 = 0.1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
y
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
x



Observations

1. Both RBF and Polynomial Kernels are able to classify the tightly-fisted structure upto a reasonably good accuracy.

2. By increasing the power p of the polynomial kernel, we get more non-linear decision boundary. This is because the higher degree polynomial is itself more non-linear. It was observed that for $p = 1$ we get a linear decision boundary, and for $p = 2$ the shape of the decision boundary resembles the quadratic curve. Thus, more the power, the better the kernel will able to curve around the tight regions.

3. The results for the RBF kernel have been plotted for two values of variance. One is $\sigma^2 = 0.1$ which is kept common across the different values of C, and the other value of variance was chosen by varying the parameter during experimentation and seeing when the best accuracy is achieved. We see that when we decrease the variance, the decision boundary curves more, and the accuracy is increased. By decreasing the variance for the RBF kernel, we get more non-linearity in the decision boundary. This is because as the variance decreases, the exponential function becomes more peaked, and more non-linear.

4. For the RBF kernel, to compare the effect of C, we look at the case of $\sigma^2 = 0.1$. The classification accuracy on the training set is equal for $C = 2500$ and $C = 500$, and it is the least for $C = 100$. The testing accuracy is maximum for $C = 2500$, followed by $C = 500$, followed by $C = 100$. This is because C is like the inverse of regularization parameter. When C is small, we are controling for the complexity, and thus we get less accuracy.

5. For the RBF Kernel, the distance between the decision boundary and the margins decrease with increasing C. This is because, when C is high, we are placing high confidence in the training examples given to us. As the problem at hand has very small margin of separation, therefore, for high C we get less margin of separation.

6. As C increases, the number of vectors in the support decrease. The number of vectors in the unbounded support increase. This is because when C decreases, the margin of separation increases and more data points lie between the margins, thus increasing the number of support vectors.