KMPA: Assignment 3 Report

Group 1: CS13B053, CS13B060, CS13B061

November 25, 2017

1 Regression

1.1 Dataset 1

The plots of underlying function, ϵ-tube, target output and approximated function obtained for Dataset 1 are documented below. The bounded and unbounded support vectors are also marked in the figure.

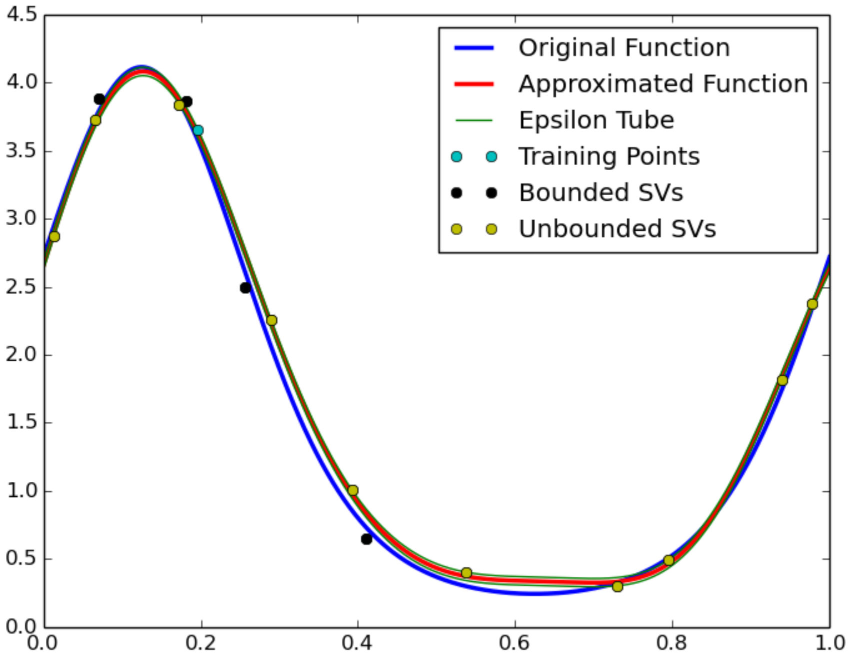

Figure 1: Approximated Function along with the ϵ-tube and the Underlying Function

Scatter Plots

The scatter plot for training data, validation data and test data for Dataset 1 are documented below.

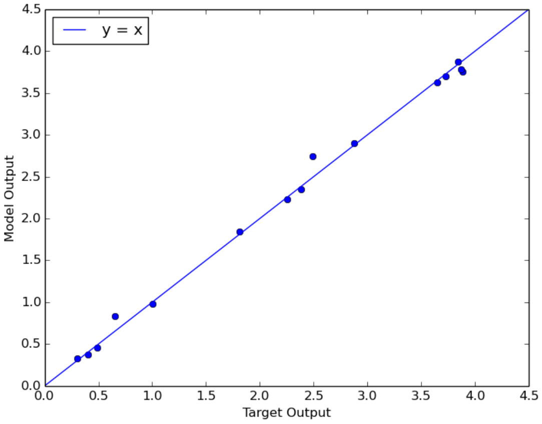

Figure 2: Scatter plot for the Training Data

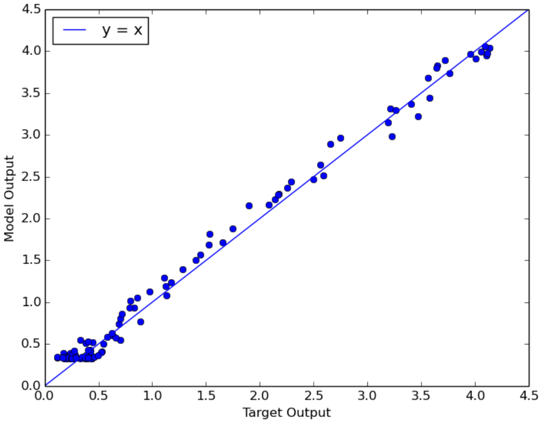

Figure 3: Scatter plot for the Validation Data

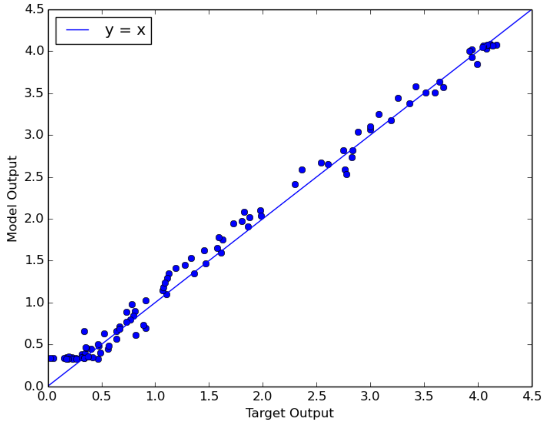

Figure 4: Scatter plot for the Test Data

Mean Squared Error vs ν Plots

The plots of Mean Squared Error (MSE) vs ν on training data, validation data and test data for Datasets 1 are documented below.

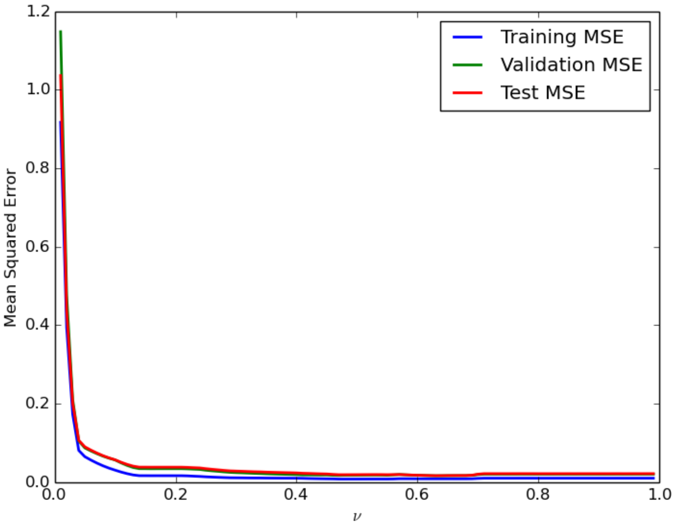

Figure 5: Mean Square Error vs ν

Model Parameters

The model was selected based on cross validation.

Training data size = 15
Validation data size = 100
Test data size = 100

Number of Support Vectors = 14
Number of Bounded Support Vectors = 4

$\epsilon = 0.030652$
$C = 10$
$\nu = 0.63$
$\gamma = 20$

Comparison with Linear Model for Regression, RBF and MLFFNN:

The decision surface and the Mean Squared Error obtained for the above models on Dataset 1 are documented below:

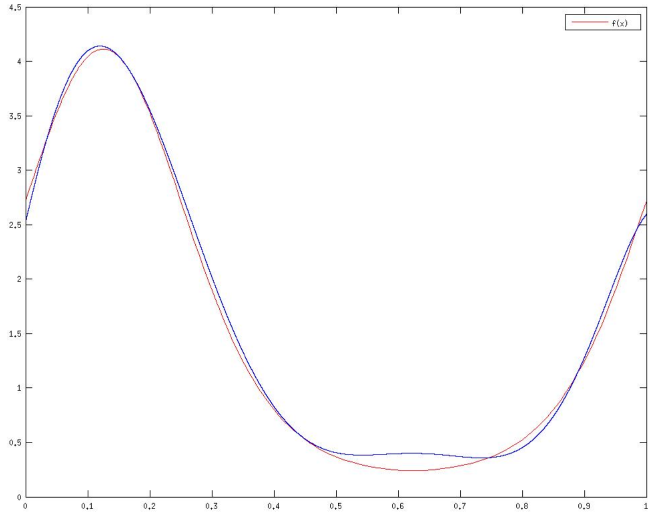

Figure 6: Decision Surface for Polynomial Curve Fitting Model on Dataset 1

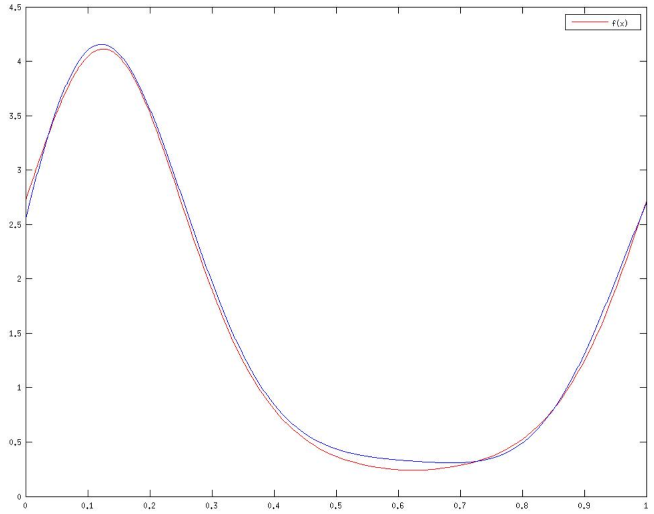

Figure 7: Decision Surface for RBF Model on Dataset 1

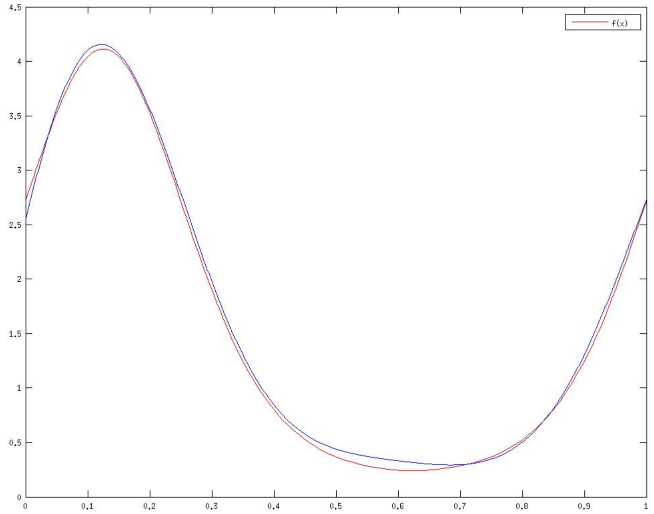

Figure 8: Decision Surface for GBF Model on Dataset 1

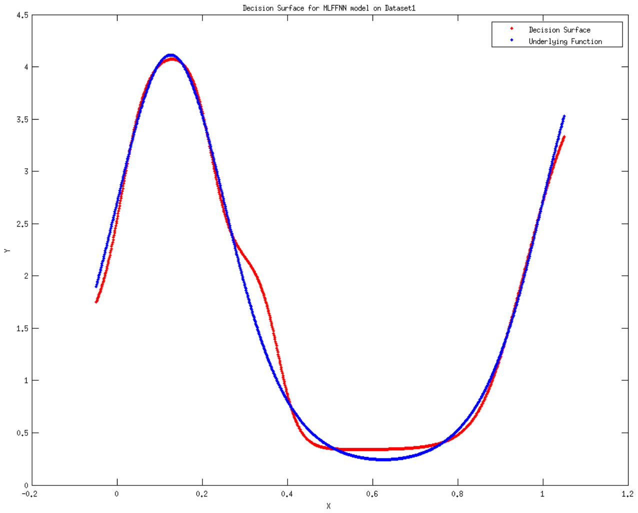

Figure 9: Decision Surface for MLFFNN Model on Dataset 1

Model	Mean Squared Error
Support Vector Regression	0.0165
Polynomial Curve Fitting	0.0168
RBF	0.0138
GBF	0.0127
MLFFNN	0.0201

Table 1: Mean Squared Error comparison of different Models for Dataset 1

Observations and Inferences:

- We see that ν-SVR approximates the underlying function very well, with the learnt model function almost lying on top of the function curve.
- Due to the small size of the training dataset, almost all the points are included as support vectors.
- SVR does better than Polynomial Curve Fitting, and comparable to RBF and GBF regression models.
- The performance of SVR is much better than that of MLFFNN at approximating the underlying function. The MSE of MLFFNN is also significantly higher.

1.2 Dataset 2

The plots of Model output and Target output for the training data, validation data and test data obtained for ν-SVR on Dataset 2 are documented below.

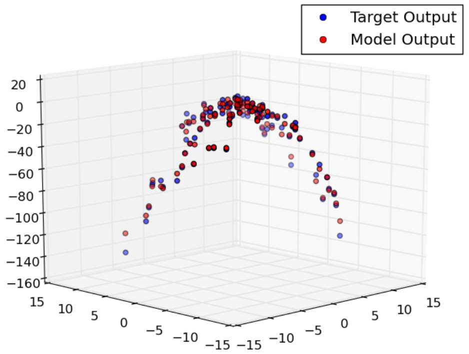

Figure 10: Model Output and Target Output for Training Data

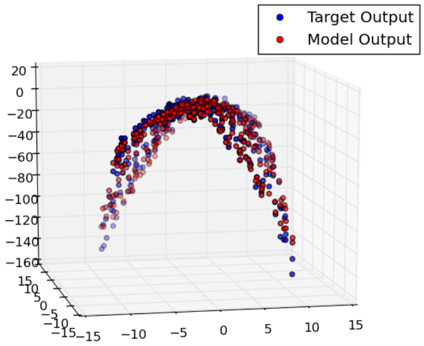

Figure 11: Model Output and Target Output for Validation Data

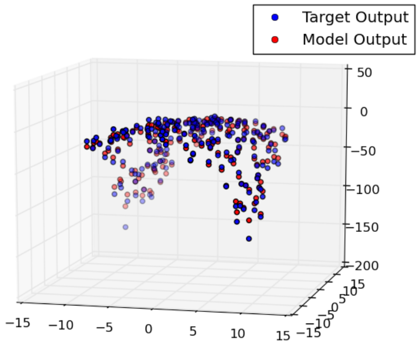

Figure 12: Model Output and Target Output for Test Data

Scatter Plots

The scatter plot for training data, validation data and test data for Dataset 2 are documented below.

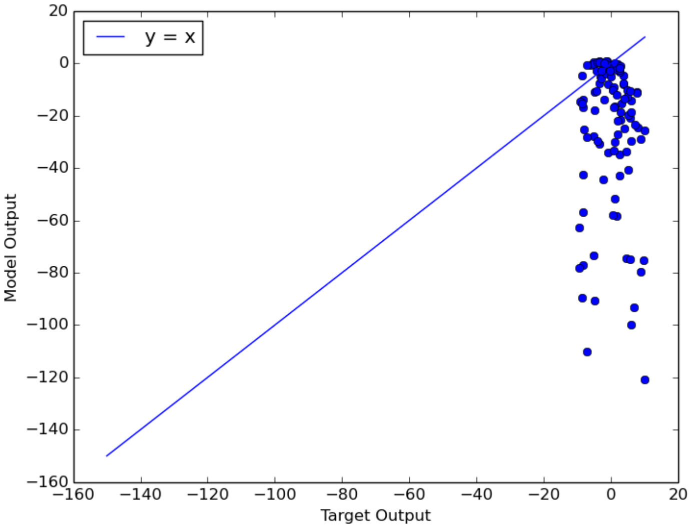

Figure 13: Scatter plot for the Training Data

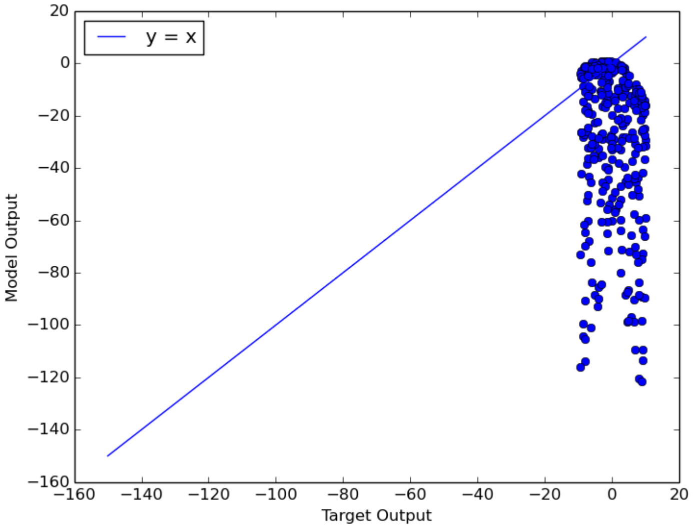

Figure 14: Scatter plot for the Validation Data

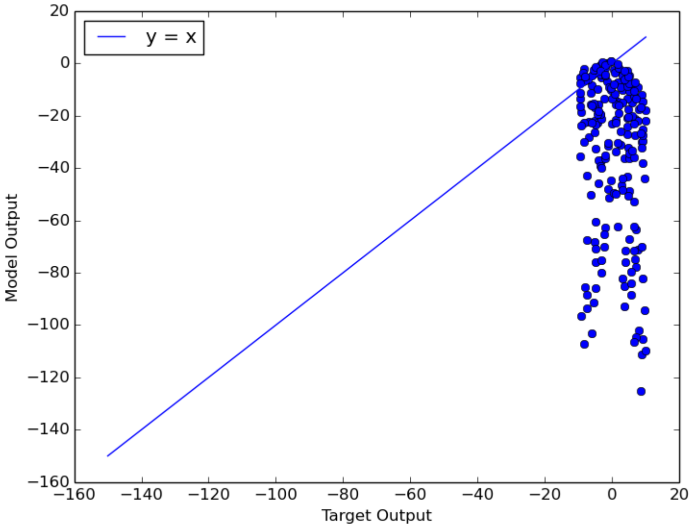

Figure 15: Scatter plot for the Test Data

Mean Square Error vs ν

The plots of Mean Squared Error (MSE) vs ν on training data, validation data and test data for Datasets 2 are documented below.

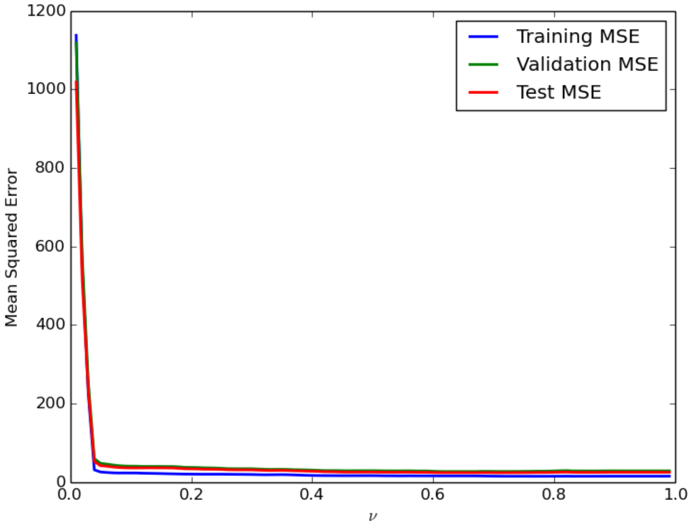

Figure 16: Mean Square Error vs ν

Model Parameters

The model was selected based on cross validation.

Training data size = 100
Validation data size = 300
Test data size = 200

Number of Support Vectors = 75
Number of Bounded Support Vectors = 58

$\epsilon = 1.289315$
$C = 100$
$\nu = 0.66$
$\gamma = 0.01$

Comparison with Linear Model for Regression, RBF and MLFFNN:

The decision surface for the above models on Dataset 2 are documented below:

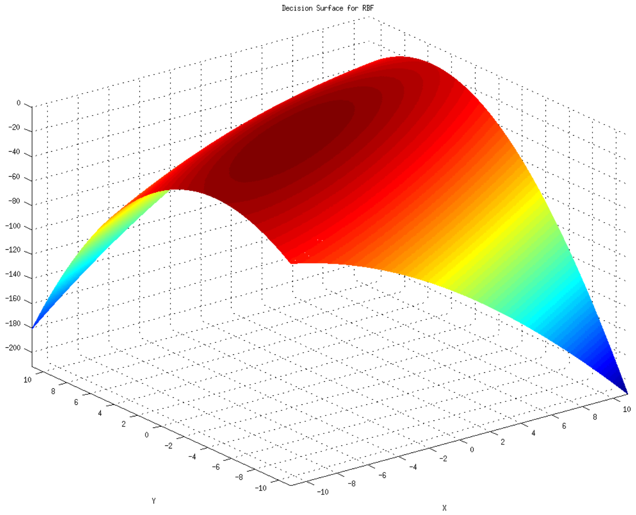

Figure 17: Decision Surface for RBF Model on Dataset 2

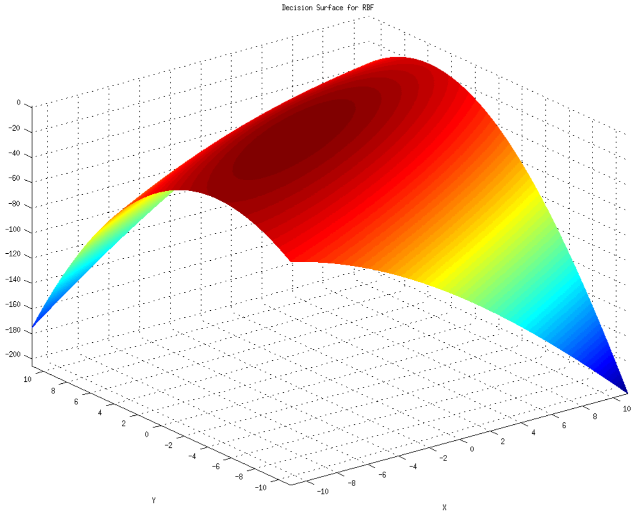

Figure 18: Decision Surface for GBF Model on Dataset 2

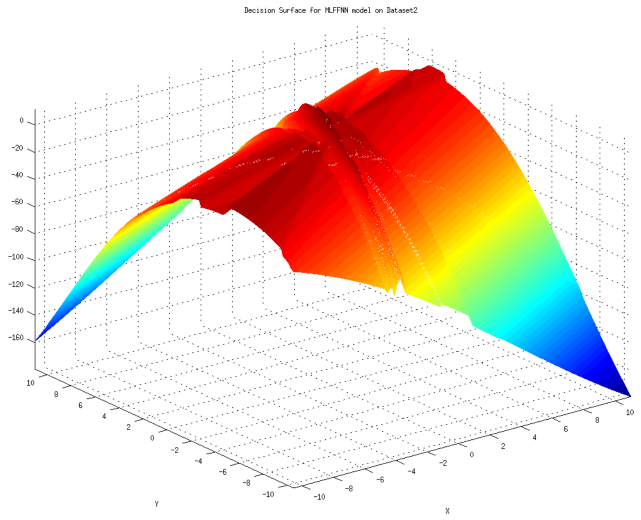

Figure 19: Decision Surface for MLFFNN Model on Dataset 2

Model	Mean Squared Error
Support Vector Regression	24.0599
RBF	13.2955
GBF	13.2856
MLFFNN	24.3123

Table 2: Mean Squared Error comparison of different Models for Dataset 2

Observations and Inferences:

- We see that SVR is able to effectively learn the underlying function from a small sized training set.
- SVR is comparable to MLFFNN.
- SVR doesn't do well compared to RBF and GBF. A reason for this could be the approximately Gaussian shaped distribution of the dataset, which might be particularly well suited for GBF and RBF regression.

2 Novelty Detection

The task of Novelty detection was done using ν-SVDD with the Gaussian Kernel for Dataset 3 and Dataset 4. We trained the model using the data points of the positive class to build the decision boundary and then used it for classifying the test points as positive or negative (novelty points). The results are documented below.

2.1 Decision Region Plot for Dataset 3

The plot of the decision region for Dataset 3 is show below. The corresponding bounded and unbounded support vectors are also marked in the given plot.

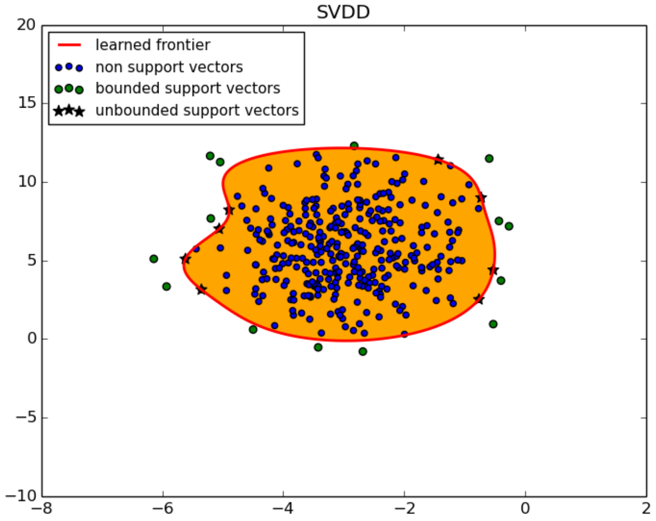

Figure 20: The Decision Region Plot with Bounded and Unbounded Support Vectors

2.2 Results

The following are the results on the test data for the two Datasets:

Dataset 3

Parameters for the best model we got are:
$\gamma = 2.1e - 05$
$\nu = 0.1$

The confusion Matrix is:

Class	1	2
1	96	7
2	4	93

Table 3: Confusion Matrix obtained for Dataset 3

True Positives:	96
False Positives:	4

Dataset 4

Parameters for the best model we got are:
$\gamma = 0.08$
$\nu = 0.05$

The confusion Matrix is:

Class	1	2
1	41	16
2	7	21

Table 4: Confusion Matrix obtained for Dataset 4

True Positives:	41
False Positives:	7

Observations and Inferences:

- We see that for Dataset 3 (2-dimensional data) the method performs extremely well. This is because except a few points, the points of the positive data class are much closer to each other and on an average are far away from the novelty data points. And hence we see very good results.

- For Dataset 4 considering the fact that it is a multi-dimensional data set (278 features), and the training of the model used the positive class data points only, the performance of the method is considerably good. If compared to the results of the classification models used on the same data set the results are comparable (refer: link).

3 Clustering

K-means clustering and Kernel K-means clustering with Guassian kernel was done on the non-linearly separable Dataset 5. The results are documented below.

3.1 K-means

The Decision region plots for the K-means clustering after various number of iterations is shown below:

After Initialization

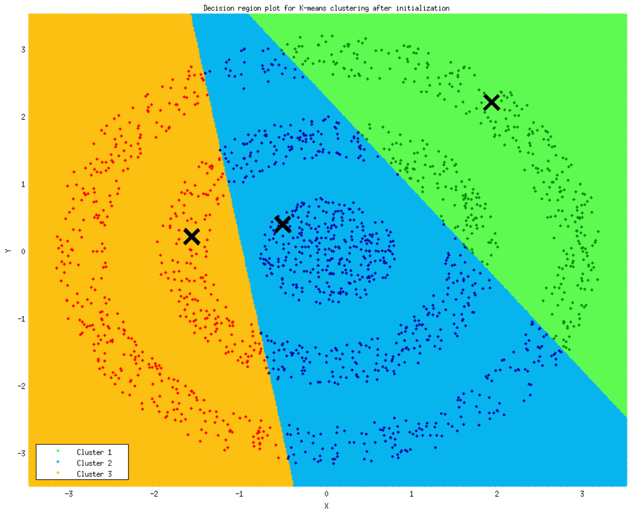

Figure 21: Decision region plot for K-means after initialization

After Second Iteration

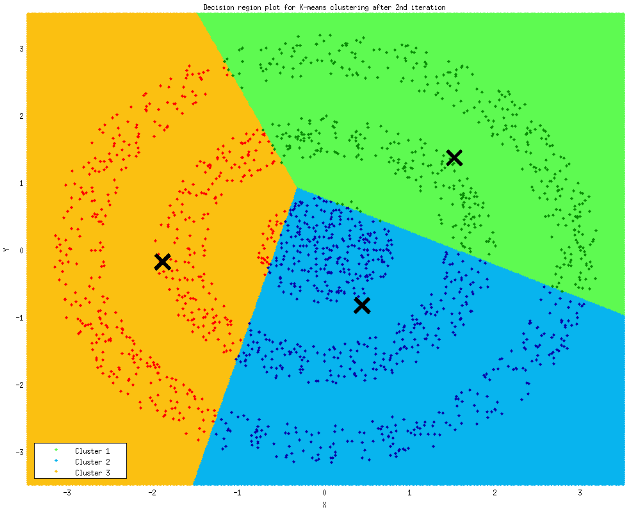

Figure 22: Decision region plot for K-means after second iteration

At Intermediate Iteration

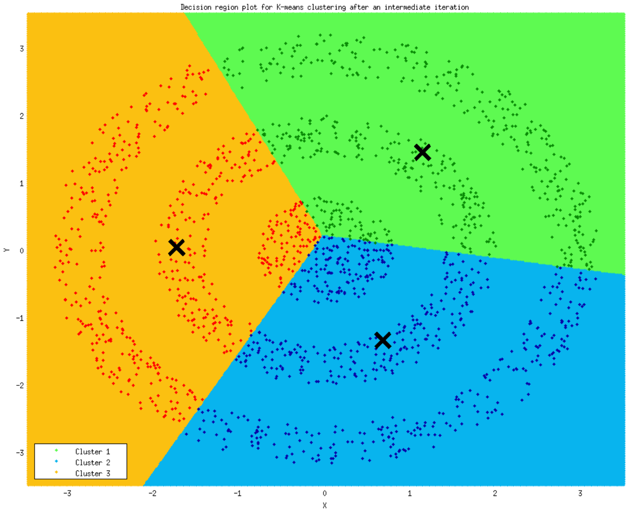

Figure 23: Decision region plot for K-means at an intermediate iteration

After Convergence

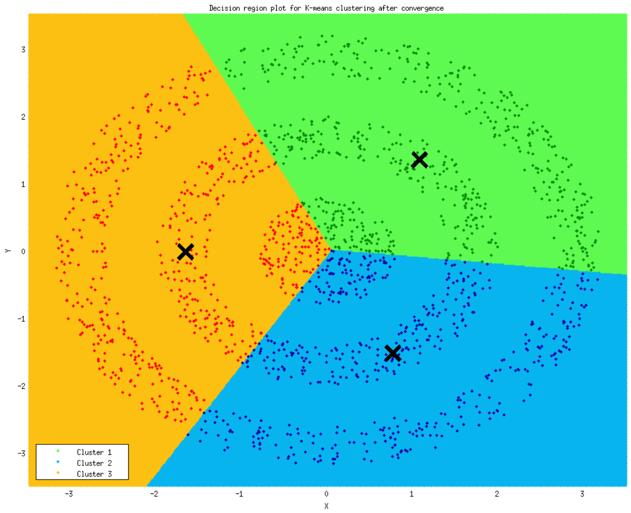

Figure 24: Decision region plot for K-means after convergence

3.2 Kernel K-means

The Decision region plots for the Kernel K-means clustering after various number of iterations is shown below:

After Initialization

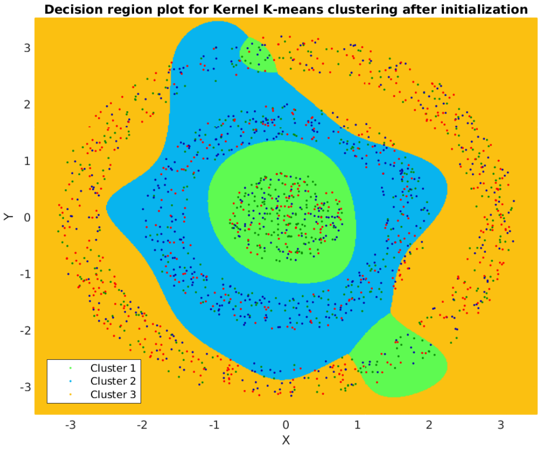

Figure 25: Decision region plot for Kernel K-means after initialization

After Second Iteration

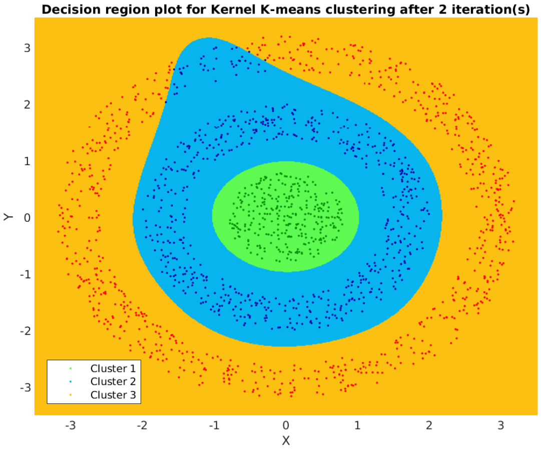

Figure 26: Decision region plot for Kernel K-means after second iteration

At Intermediate Iteration

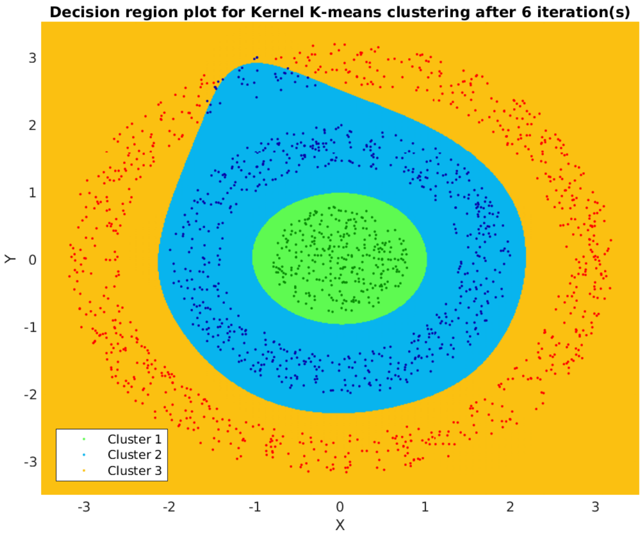

Figure 27: Decision region plot for Kernel K-means at an intermediate iteration

After Convergence

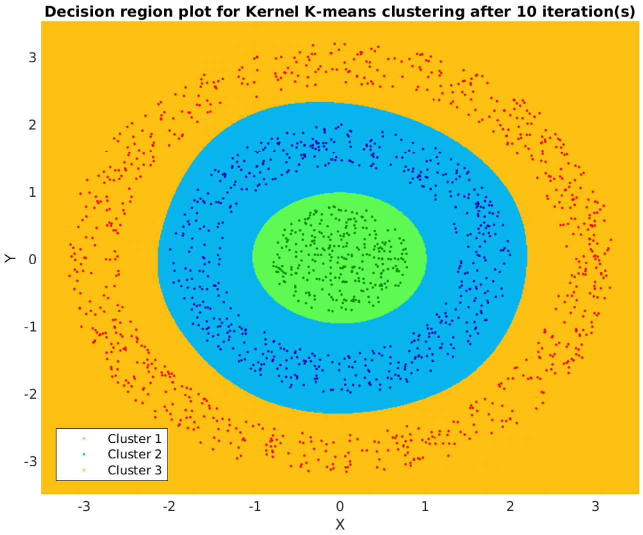

Figure 28: Decision region plot for Kernel K-means after convergence

3.3 Observations and Inferences:

- K-means works well when we have compact convex clusters. In our case we have concentric rings which do not in any way makes compact clusters. Hence K-means as expected does not perform well.
- In Kernel K-means, if the indicator variables are initialized completely randomly then we see that this method clusters the inner class well, but fails to separate outer rings. But if we initialize the cluster labels of some of the points correctly and incorrectly for other point, then the Kernel K-means clustering performs well and separates all the three clusters well.

4 Semi Supervised Learning

4.1 Dataset 6

Test accuracies and parameters of the best models found for the four methods are documented below:

Supervised ν-SVM

Test Accuracy: 20 / 37

Parameters:

- ν: Any
- Kernel: Any

Self-trained ν-SVM

Test Accuracy: 35 / 37

Parameters:

- ν: 0.01
- γ: 20
- Kernel: RBF

Note: For the experiment we considered those points that were 0.95 margin away from the separating hyper plane to be confidently labelled points.

Graph Based Label Propagation

Test Accuracy: 37 / 37

S^{3}VM

Test Accuracy: 35/37

Parameters:

- γ: 450
- Kernel: RBF

Decision region plots obtained for Self-training with ν-SVM, Graph based semi-supervised method using Label propagation, Semi-supervised SVM and supervised ν-SVM for Dataset 6 are documented below:

Supervised ν-SVM

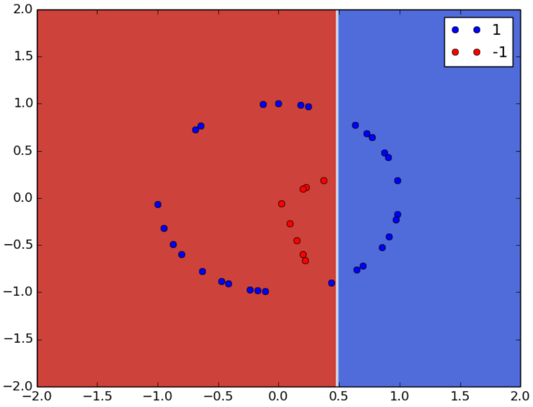

Figure 29: Decision Region Plot for Supervised ν-SVM

Self-trained ν-SVM

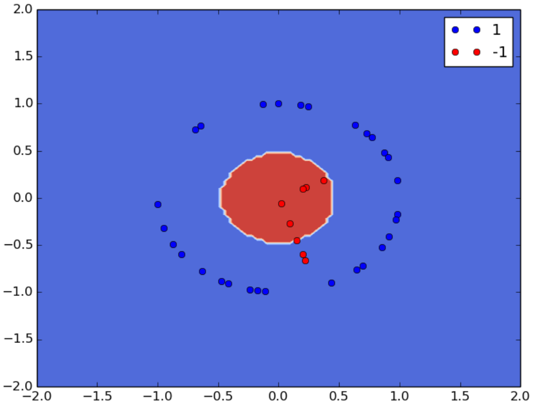

Figure 30: Decision Region Plot for Self Trained ν-SVM

Graph Based Label Propagation

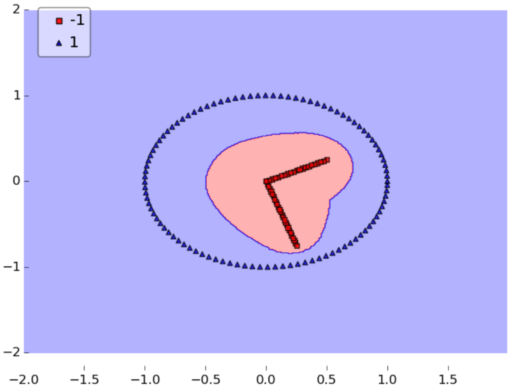

Figure 31: Decision Region Plot for Graph Based Semi-Supervised method using Label Propagation

S^{3}VM

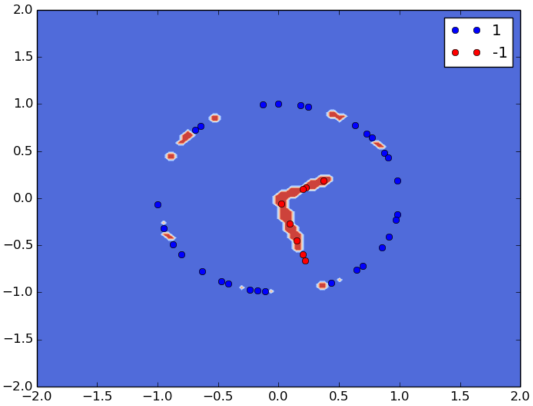

Figure 32: Decision Region Plot for Semi-Supervised SVM

4.2 Dataset 7

Test accuracies and parameters of the best models found for the four methods are documented below. Total number of points in the test set was 393.

Supervised ν-SVM

The test accuracy and the parameters of the best model are tabulated for different percentages of labelled data.

Percentage	1	10	20	50	80	100
Test Accuracy	204	198	210	248	266	279
ν	0.01	0.01	0.01	0.01	0.03	0.01

Kernel: RBF
Gamma: 1

Self-Trained ν SVM

The test accuracy and the parameters of the best model are tabulated for different percentages of labelled data.

Percentage	1	10	20	50	80	100
Test Accuracy	204	198	210	248	266	279
ν	0.01	0.01	0.01	0.01	0.03	0.01

Kernel: RBF
Gamma: 1

Note: For the experiment we considered those points that were 1.0 margin away from the separating hyper plane to be confidently labelled points.

Graph Based Label Propagation

For determining the graph we used KNN method with $k = 4$ to decided the connections of the graph. We also tried fully connected graph but the performance was better for the KNN based method. The test accuracy for the best model are tabulated for different percentages of labelled data.

Percentage	1	10	20	50	80	100
Test Accuracy	256	340	357	381	385	385

S^{3}VM

The test accuracy and the parameters of the best model are tabulated for different percentages of labelled data.

Percentage	1	10	20	30	50	80	100
Test Accuracy	228	223	258	284	218	210	211

Kernel: RBF
Gamma: 1

Observations and Inferences:

- For Dataset 6 and Dataset 7, we can see that the Graph based label propagation method works the best. It can be the case that the labels are “smooth” with respect to the graph, such that they vary slowly on the graph. That is, if two instances are connected by a strong edge, their labels tend to be the same.

- For Dataset 6 both self-trained ν-SVM and S^{3}VM gives similar results which are better than the supervised ν-SVM. Hence we see that for this dataset the semi-supervised method works better than the supervised method (ν-SVM). Hence the unsupervised methods are using the unlabelled data efficiently.

- For Dataset 7, both self-trained ν-SVM and supervised ν-SVM gives very similar results. This may be because the self-trained ν-SVM might not be able to find sufficient confident data points and hence is not able to use the unlabelled data effectively.

- For Dataset 7, for S^{3}VM we see a very different trend. First, the performance increases as the labelled data increases and then it decreases. The best performance for S^{3}VM was obtained when 30% of the data was labelled and it was better than the best performance of both self-trained ν-SVM and supervised ν-SVM. The assumption of S^{3}VMs is that the classes are well-separated, such that the decision boundary falls into a low density region in the feature space, and does not cut through dense unlabeled data. If this assumption does not hold, an S^{3}VM's search for a gap between the two classes may stuck in one of many possible local minima. The resulting decision boundary may be worse than the decision boundary of an SVM that does not try to exploit unlabeled data at all.

5 Classification using kernels for Structured Data

This task is to better understand the power of kernel methods, by performing classification on structured data (text documents), without any need to know the feature vector representation. This is done by defining a kernel for each pair of text documents (inner product in feature space).

Dataset Used:

We have used subset of the popular **Reuters-21578** dataset for performing Text Classification task. The Reuters-21578 dataset contains documents that appeared in the Reuters newswire in 1987. The dataset contains several categories of articles. We have used four categories (earn, trade, crude and acq) of articles from the dataset for performing our classification task.

Kernel Used:

The standard approaches for Text Classification maps the document to a high dimensional feature vector where each entry of the vector represents the presence or absence of a feature. This approach loses all the word order information only retaining the frequency of the terms in the document.

We used **String Subsequence Kernel** for the task, which captures the word order information. This approach considers the documents as symbol sequences. The feature space in this case is generated by the set of all (non-contiguous) substrings of k-symbols. The more substrings two documents have in common, the more similar they are considered (the higher their inner product).

The main idea of the String Subsequence Kernel is to compare the two documents (or strings) by means of the substrings they contain: the more substrings in common, the more similar they are. Substrings do not need to be contiguous, and the degree of contiguity of one such substring in a document determines how much weight it will have in the comparison.

There are two hyper-parameters in this kernel, subsequence length k and a decay factor $\lambda \in (0,1)$. Then for two strings s and t, the kernel is defined as:

$$\mathcal{K}(s,t) \;=\; \sum_{u\in\Sigma^k} \sum_{\boldsymbol{i}:u=s[\boldsymbol{i}]} \sum_{\boldsymbol{j}:u=s[\boldsymbol{j}]} \lambda^{l(\boldsymbol{i})+l(\boldsymbol{j})} \tag{1}$$

where Σ^k is the set of all substrings of length k, $\boldsymbol{i}$ is the vector of indices such that $u_j = s_{i_j},\ j = 1,\ldots,|u|$, similarly the vector $\boldsymbol{j}$ for string t, and the length $l(\boldsymbol{i})$ of the subsequence in s is $i_{|u|} - i_1 + 1$.

Package Used:

Harry - A Tool for Measuring String Similarity, was used to this task. Given the documents the package returns the kernel gram matrix. Then using the kernel gram matrix, a Support Vector Classifier was trained.

Results

The experiment was performed for different values of the hyper-parameters k and λ, and the accuracy obtained are shown in Table 5.

Length k	Decay Factor	Accuracy (%)
2	0.01	89.25
	0.1	90.00
	0.5	91.00
	0.9	93.75
	0.95	**94.25**
	0.99	93.75
3	0.01	86.00
	0.1	86.75
	0.5	86.50
	0.9	88.75
	0.95	88.75
	0.99	89.50
5	0.01	62.25
	0.1	63.00
	0.5	70.25
	0.9	68.25

Table 5: Accuracy for different hyper-parameters for Text Classification on Reuters dataset

Observations

- Using the String Subsequence Kernel gives very good accuracy (94.25%) for the classification of the documents of the Reuters dataset.
- This approach captures more information about the document than the normal bag of words classification for texts. Mainly it captures the word order which is disregarded by the bag of words model.
- This kernel does not use any domain knowledge, in the sense that it considers the document just as a long sequence.