Master of Artificial Intelligence

Support Vector Machines Course Report

Name : Keerthitheja Samudrala Chandrasekar
Student Number : r0773368

Prof. Johan Suykens
Artificial intelligence
KU Leuven

Contents

1. Exercise 1: SVM Classification

1.1 Exercise 1

1.1.1 Two Gaussians :

Question: *Obtain a line to classify the data by using what you know about the distributions of the data. In which sense is it optimal ?*

There are lots of possibilities to draw a line that can separate these two data sets. But, selecting an optimal line among those possibilities is a challenge. A threshold can be fixed and the line can be drawn by calculating the midpoint between the closest points of the two classes. But, when outliers occur, this threshold based technique can give a lot of misclassifications. Therefore, for our problem we use Support Vector Classifier that can generate an optimally best hyperplane to separates the two data sets. It is also known as soft margin classifier. The shortest distance between two data points of the different classes gives the margin. For a N- dimensional data, the classification is done using Support Vector Machines and it gives a best hyper plane in N-dimensional space that separates all the classes from one another. Since, our problem has a 2-Dimensional data, our Support Vector Classifier is a line.

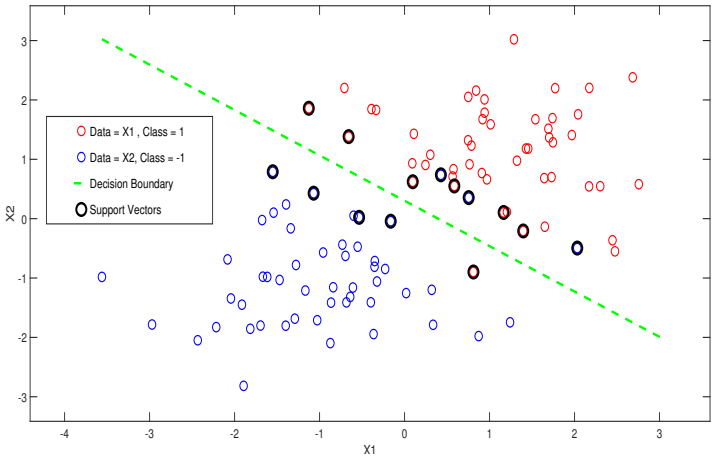


Figure 1.1: Classification

SVM uses few data points from the training data as support vectors that determines the weights and position of hyper planes. The below equation define the hyper plane of a SVM.

$$f(x) = x\beta + b \tag{1.1}$$

where β is the linear predictor coefficient and b is the bias term. These β and b terms are the optimal values of a hyper plane learnt by the classifier as SVM tries to maximize the margin i.e., distance between the hyper plane and support vectors. This is done by solving the Lagrange equation using optimization techniques. We train the SVM classifier on our input data set using *fitcsvm* in Matlab. Hence, the separating line (in our case the green line in the figure 1.1) that is drawn using the values of β and b is an optimal one that allows only a few misclassifications as shown in the figure 1.1.

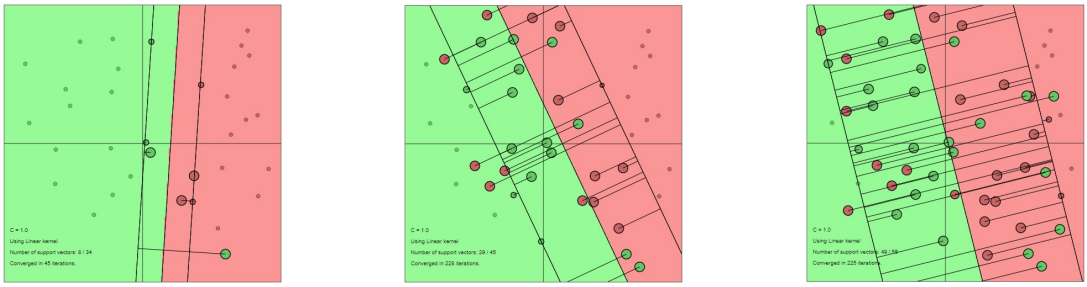
1.1.2 Support Vector Machine Classifier:

When the data points of two classes are linearly separable then SVM tries to classify these data points with a line in such a way that the separation between these two classes is maximum. As we know from solving the Lagrangian equation, the solution must satisfy the following the relations

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i, \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{1.2}$$

Solving the above equation 1.2 using dual problem we get α_i.

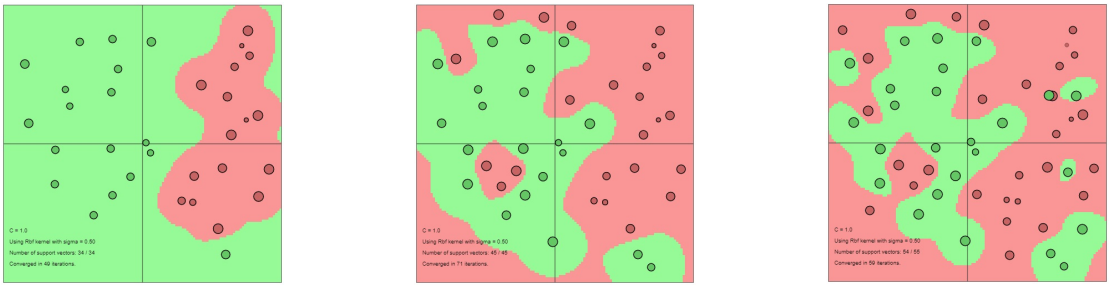
The below figures 1.2 and 1.3 shows SVM classifier with linear and RBF kernels. In figure 1.2a and 1.3a it can be seen that the data points are on the right side of their respective classes. There is no much change in the decision boundary even after adding more points on the same side of the class. Both linear and RBF kernels produce same classification output.



(a) Points on right side (b) Points on wrong side (c) More points on wrong side

Figure 1.2: SVM Classification with Linear Kernel

In figures 1.2b and 1.3b few points of one class are added on the wrong sides of hyper plane. In this case, the new points influence the decision boundary and more number of non-zero elements of α_i occurs from equation 1.2. Also, the soft margin increases as seen in fig 1.2b. Since, the data points are not linearly separable, the SVM classifier with linear kernel takes more number of support vectors (elements with $\alpha_i \neq 0$) into account while deciding a decision boundary. This accounts for few misclassifications. When it comes to the classifier with RBF kernel in fig 1.3b, it classifies all the data points correctly with no misclassifications.



(a) Points on right side (b) Points on wrong side (c) More points on wrong side

Figure 1.3: SVM Classification with data points on wrong side of a hyperplane

In figures 1.2c and 1.3c more points of one class are added on the wrong sides of the hyper plane. Wrong data points affect the decision boundary to a great extent with linear kernel and the soft margin becomes wide in fig 1.2c . It can be observed in figure 1.2c that even though linear kernel takes almost every point as a support vector, it still fails to give a good decision boundary and a lot of misclassifications occur. This means, changing the support vectors can change the decision boundary. But the classifier with RBF kernel, classifies all the data points correctly with only one misclassification as seen in fig 1.3c.

Try out different values of the regularization hyperparameter C and the kernel parameter sigma. What is the role of the parameters? How do these parameters affect the classification outcome?

The parameter C is responsible for optimization. It tells the SVM classifier on how much misclassification it can allow on a given data set. For some larger value of C, the SVM classifier

will give us a hyperplane with smaller margin and classify all the training data points correctly. Conversely for some smaller value of C, the SVM will give us a hyperplane with wider margin and allows some misclassifications even if the training data is linearly separable.

Sigma determines the reach of a training data point. For a smaller value of sigma, the SVM classifier reaches to the training points that are further away from them. This can lead to over-fitting of the model. For larger values of sigma, the model looks for points that are closer and it will be able to generalize better. The effect of these hyper parameters C and sigma is explained in detail in the below question.

Compare classification using the linear kernel with classification using the RBF kernel. Which performs better? Why?

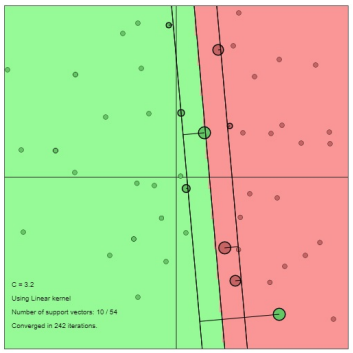
As seen from figure 1.2a and 1.3a, SVM with both linear and RBF kernel perform in the same manner as the data points are linearly separable. But from figures 1.2b, 1.3b, 1.2c, 1.3c we can observe that RBF out performs the linear kernel when the data is non-linearly separable. What RBF does is, it creates a non-linear combination of the features to project the data points onto a higher-dimensional space and uses a linear decision boundary to separate the classes(kernel trick). Training a SVM with linear kernel is faster when compared to RBF kernel. Therefore, linear kernel can do the job for linearly separable data points. But, for non-linearly separable data RBF would be a better choice

Question: *What is a support vector? When does a particular datapoint become a support vector ? When does the importance of the support vector change? Illustrate visually.*

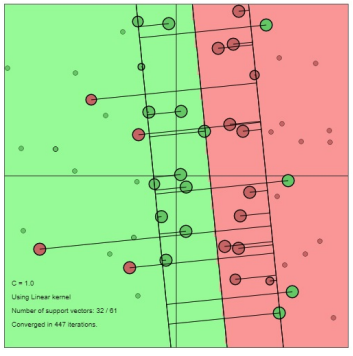
Support vectors are nothing but the training data points which lie on(or inside of) the margin line. SVM tries to find an optimal hyper plane $y(x) = w^T x_i + b$ such that for an unknown value of x, it gives the class of x. Therefore, its goal is to estimate w. As given in equation 1.2 when we solve the Lagrangian dual of SVM, we get $w = \sum_{i=1}^{N} \alpha_i y_i x_i$. Substituting the value of w in $y(x)$ gives the below equation.

$$y(x) = \sum_{i=1}^{N} \alpha_i y_i x_i^T x + b \tag{1.3}$$

Here α_i are the variables of duality. A data point becomes a support vector when the α_i value is non-zero and the point lie on or inside the margin line. Equation 1.3 denotes the sparse representation of SVM. It can be seen from eq 1.3 that our linear classifier is represented using only the points that lie on or inside the margin line. With this we can now transform the SVM into a non-linear classifier using the kernel trick and compute classification hyperplanes in large dimensional spaces as y(x) is now only a dot product of terms. From figure 1.4, it can be seen that in the case of linearly separable points (fig 1.4a), the support vectors are fewer in number and the soft margin has a smaller width. The one



(a) Linearly Separable Points

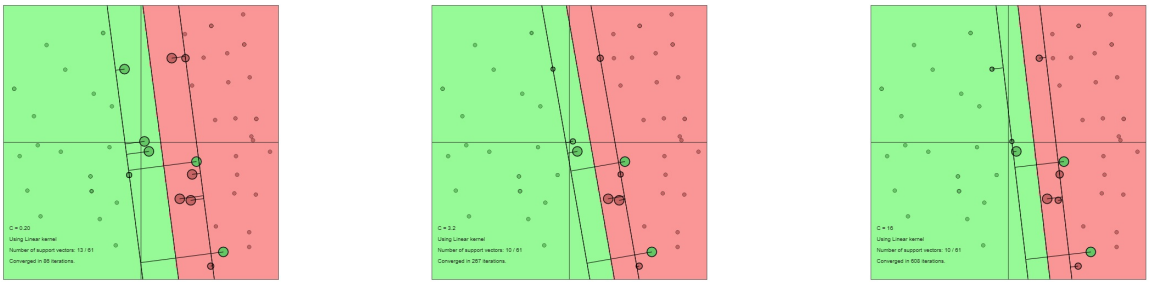


(b) Non-linearly Separable Points

Figure 1.4: Support Vectors during classification

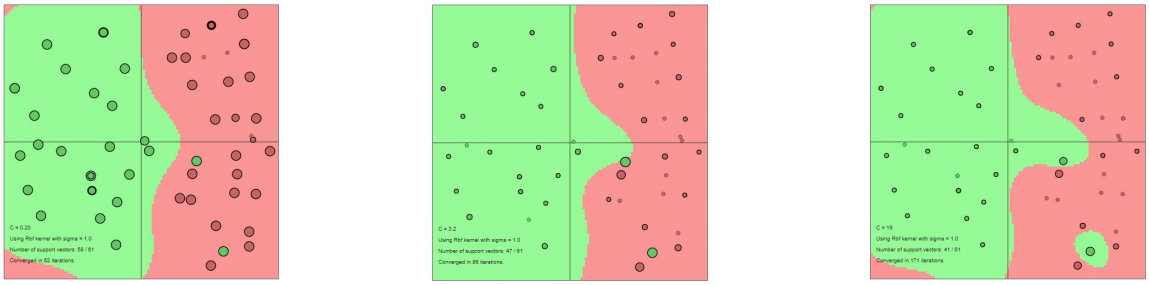
misclassified green point is also considered as a support vector. Whereas, in the case of non-linearly separable points (fig 1.4b) every misclassified point is considered as a support vector and SVM takes lot more points in arriving at a decision boundary. Also, the width of the soft margin is huge in figure 1.4a compared to the one in figure 1.4a. Hence, we can say that the support vectors are responsible for the position and orientation of a hyper plane.

Question *What is the role of parameters C and sigma? What happens to the classification boundary if you change these parameters. Illustrate visually.*
The regularization parameter C controls the misclassifications done by SVM classifier. The effect of C is illustrated in the following figures.



(a) C = 0.2 (b) C = 3.2 (c) C = 16

Figure 1.5: SVM with Linear Kernel and change in C values



(a) C = 0.2 (b) C = 3.2 (c) C = 16

Figure 1.6: SVM with RBF Kernel and change in C values

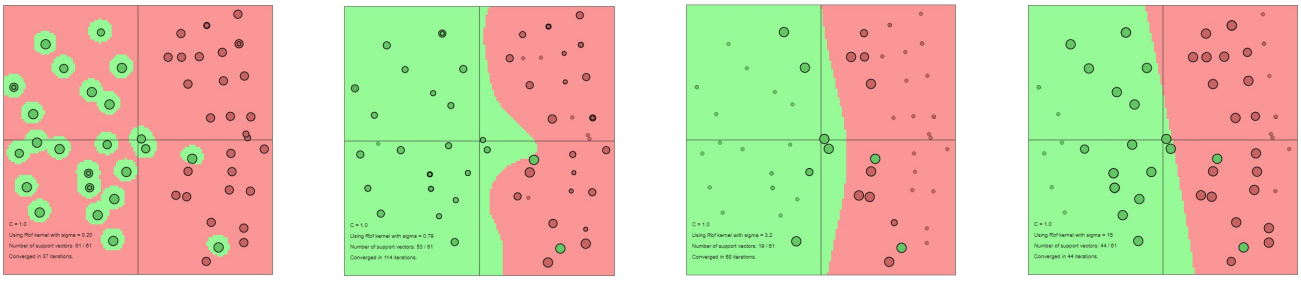
From figure 1.5c, it can be seen that for a value of C=16, the decision boundary for a linear kernel has smaller margin and less number of support vectors. For the same value of C in figure 1.6c, the RBF kernel correctly classifies all the training points. As the C values is reduced to 0.2 and 3.2 in figures 1.5a and 1.5b, the linear kernel uses more support vectors and also accounts to misclassifications. The soft margin and become wider for lesser values of C. Similarly for the RBF kernel in figures 1.6a and 1.6b, misclassifications occur for smaller value of C. Therefore, from figures 1.5 and 1.6, it can be said that the a larger of C makes the model over-fit while a smaller value of C makes the model under fit.

The RBF kernel for SVM classifier in dual space is defined as below

$$k(x, x_k) = exp\left(- \frac{||x - x_k||_2^2}{\sigma^2} \right) \tag{1.4}$$

From the above equation 1.4, it can be seen that the value of σ influences the distance between x and x_k. The value of $k(x, x_k)$ tends to zero when σ is much smaller than $||x - x_k||_2^2$. This can makes the classifier reach points that are further away which leads to over fitting. As the value of σ increases the classifier will look for points that are closer and hence it will generalize better.

In figure 1.7 it can be seen that for a smaller value of $\sigma = 0.2$ the classifier tries to fit all the



(a) Sigma = 0.2 (b) Sigma = 0.79 (c) Sigma = 3.2 (d) Sigma = 16

Figure 1.7: SVM with RBF Kernel and change in sigma values

data points which indicates over fitting. As, the value of σ increase to 0.79, 0.32 and 16, the model generalizes better.

Question: *What happens to the classification boundary when sigma is taken very large? Why?*

As seen from figure 1.7d, for a higher value of $\sigma = 16$ the model generalizes better and gives almost a linear decision boundary. This means that the radius of influence of support vectors will include only the support vectors.

1.1.3 Least-Squares Support Vector Machine Classifier:

Influence of hyperparameters and kernel parameters

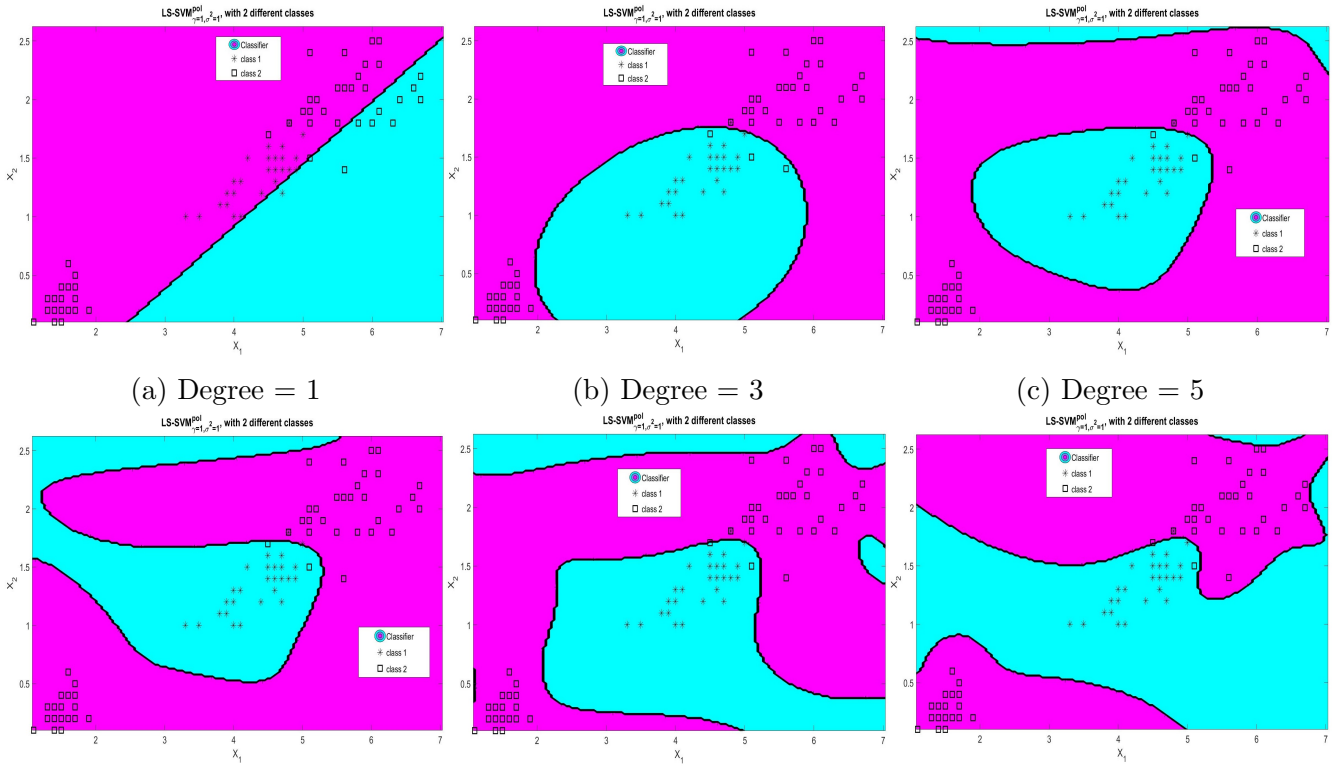
Try out a polynomial kernel with degree = 1; 2; 3; ... and t = 1 (fix gam = 1). Assess the performance on the test set. What happens when you change the degree of the polynomial kernel?

Degree	Misclassifications	Error Rate on test set (%)
1	11	55.00
3	0	0
5	0	0
7	0	0
9	0	0
11	0	0

Table 1.1: Error rate of polynomial kernel on test set

From the figure 1.8 and table 1.1, it can be observed that for degree = 1, the decision boundary is a linear one and lot of misclassifications occur on the training data points. Even on test set the classifier gives 11 misclassifications with a error rate of 55 %. As the degree value gets increased, the model performs better and the number of misclassifications gets reduced. The classifier gives exact predictions on the test sets for degree values greater than 1. Also, for the polynomial kernels with degree 9 and 11, the model tries to cover all the training samples with complex decision boundaries which indicates that it is moving towards over-fitting. Therefore, a good range of degree values can be between 3 and 9.

Try out a good range of different sig2 values as kernel parameters (fix gam = 1). Assess



(a) Degree = 1 (b) Degree = 3 (c) Degree = 5

(d) Degree = 7 (e) Degree = 9 (f) Degree = 11

Figure 1.8: SVM with polynomial Kernel and change in degree values

the performance on the test set. What is a good range for sigma. Fix a reasonable choice for the sig2 parameter and compare the performance using a range of gam. What is a good range for gam?

As seen from the figure 1.9 and table 1.2, the classifier works good on the test set for a range of σ^2 between 0.1 and 10 when the value of γ is fixed to 1. If σ^2 is increased further, the model performance decreases and gives lot of misclassifications.

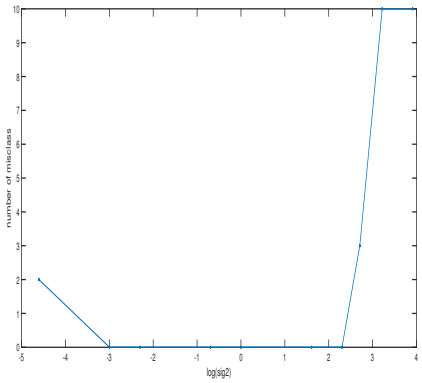


Figure 1.9: $\log(\sigma^2)$ VS misclassifications

Gamma	Sigma	Mis-classifications	Error Rate on test set (%)
1	0.01	2	10
1	0.05	0	0
1	0.1	0	0
1	0.5	0	0
1	1	0	0
1	5	0	0
1	10	0	0
1	15	3	15
1	25	10	50
1	50	10	50

Table 1.2: Performance of SVM for various σ^2 values

SVM classifier is sensitive to changes in γ. As, seen from figure 1.10 and table 1.3, the model performs poorly for lower values of γ and gives stable results for higher values. This means that for very small values of γ, the model fails to capture the complexity of the data. For larger values of γ, even though the model learns complex shapes, it focuses on smoothing the curve rather than minimizing the errors. This could result in poor performance of model without proper fitting of data points. A good range of γ can be between 1 to 5, when σ is fixed to 2.

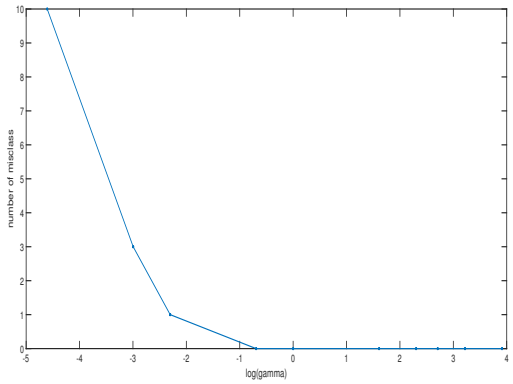


Figure 1.10: log(γ) VS misclassifications

Sigma	Gamma	Mis-classifications	Error Rate on test set (%)
2	0.01	10	50
2	0.05	10	50
2	0.1	3	15
2	0.5	1	5
2	1	0	0
2	5	0	0
2	10	0	0
2	15	0	0
2	25	0	0
2	50	0	0

Table 1.3: Performance of SVM for various γ values

Similar results as described above is seen with the script SampleScript_Iris.m

Tuning parameters using validation

We continue with the Iris data set and tune parameters of γ and σ^2 for a range of values. We use the range of γ as (0.01; 1; 5; 25; 100; 1000) and σ^2 as (0.001; 0.1; 0.5; 1; 5; 10; 25; 50; 100; 10000). Unlike in the previous sections, we now split the training data randomly into 80% as training and 20% as validations sets. The SVM classifier is trained on the training data and the results are shown on the validation set. Figure 1.12 shows the performance error (%) of SVM for a fixed γ and a range of σ^2. Figure 1.11 shows a 3D plot over range of (γ, σ^2) and performance errors. It is observed from figure 1.12 that the performance error varies between 20 to 40 percent for the given range of (γ, σ^2). Using these results it is difficult to determine a best performing γ and σ values for the random split technique. Another way to determine a good range of (γ, σ^2) is using k-fold cross validation and leave-one-out validation.

In k-fold cross validation the training data is split into k subsets and the model is trained with k-1 sets of data and the kth set is used for validation. This is repeated over all the k subsets of data in a loop. Next we perform leave-one-out method similar to k-fold cross validation. The result of these methods is not performance rate but rather a average of errors over number of iterations of k-sets. The results are shown in figures 1.14 and 1.16. It can be seen that the performance for both k-fold and leave-one-out methods are nearly same. Therefore, in case of larger data sets, k-fold method is preferred as it is computationally faster than leave-one-out and for smaller data sets leave-one-out is preferred. From figure 1.11, it can be seen that random split validation method exhibits high variance compared to k-fold and leave-one-out methods in figures 1.13 and 1.15 giving a clear indication of best (purple color) and least performing (pale yellow and cyan) γ and σ^2 values.

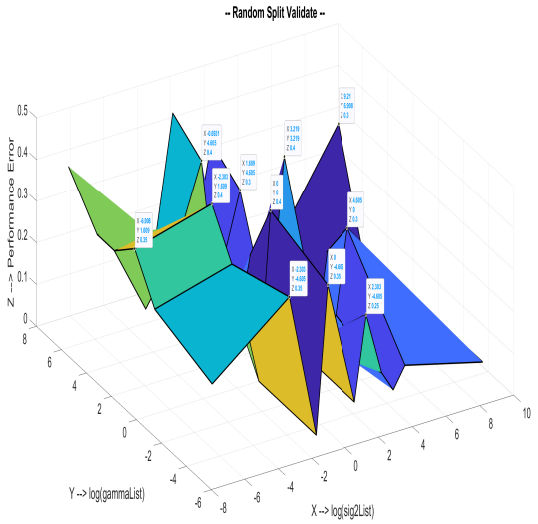
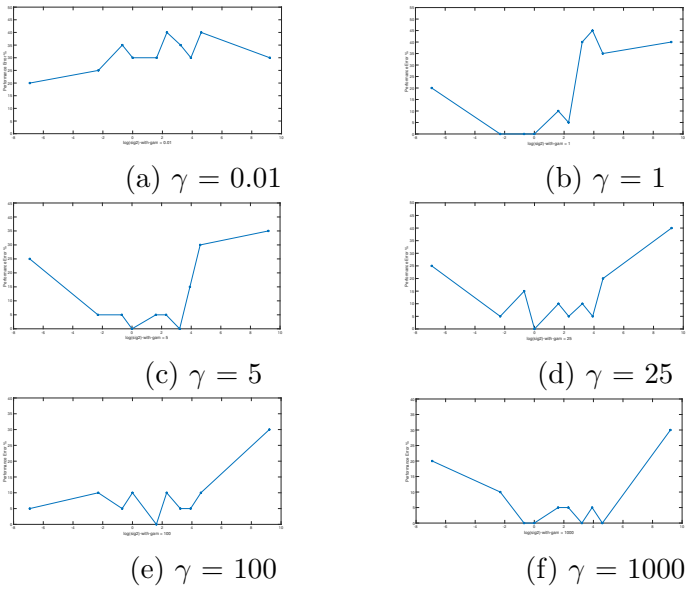


Figure 1.11: Random Split Validation - ($\log(\sigma^2)$ vs log (γ) vs Performane Error)



(a) $\gamma = 0.01$ (b) $\gamma = 1$ (c) $\gamma = 5$ (d) $\gamma = 25$ (e) $\gamma = 100$ (f) $\gamma = 1000$

Figure 1.12: Iris data : Random Split Validation ($\log(\sigma^2)$ values for each level of γ Vs Performance Error) (%) on validation set.

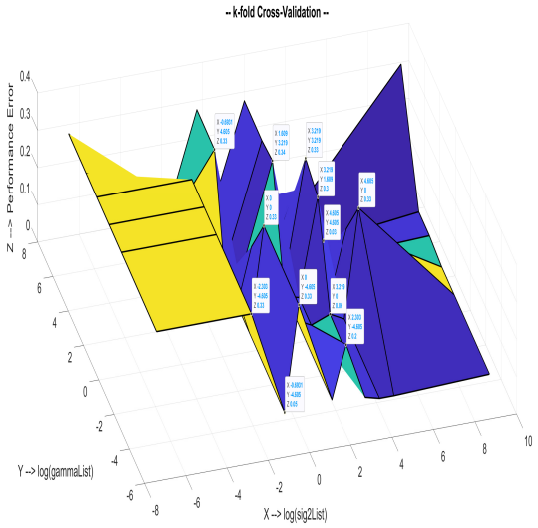
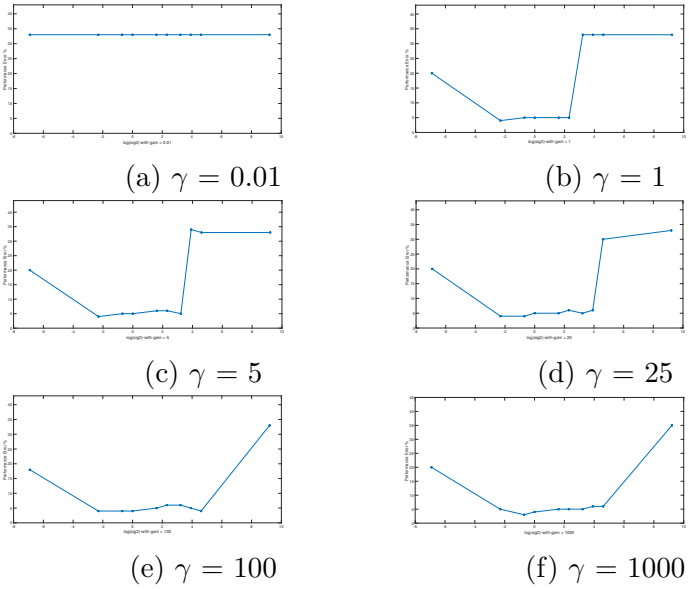


Figure 1.13: k-fold Cross Validation - ($\log(\sigma^2)$ vs log (γ) vs Performane Error)



(a) $\gamma = 0.01$ (b) $\gamma = 1$ (c) $\gamma = 5$ (d) $\gamma = 25$ (e) $\gamma = 100$ (f) $\gamma = 1000$

Figure 1.14: Iris data : k-fold CV ($\log(\sigma^2)$ values for each level of γ Vs Performance Error) (%)

This can also be seen from the smoothness of curves in the case of k-fold and leave-one-out.

From figures 1.14 and 1.16 we can choose the optimal value of (γ, σ^2) that gives the least performance error. Some good combinations of (γ, σ^2) can be as shown in table 1.4.

γ	σ^2
25	(0.1, 0.5, 1, 5)
100	(0.1, 0.5, 1, 5)
1000	(0.5, 1, 5)

Table 1.4: Optimal (γ, σ^2) combinations

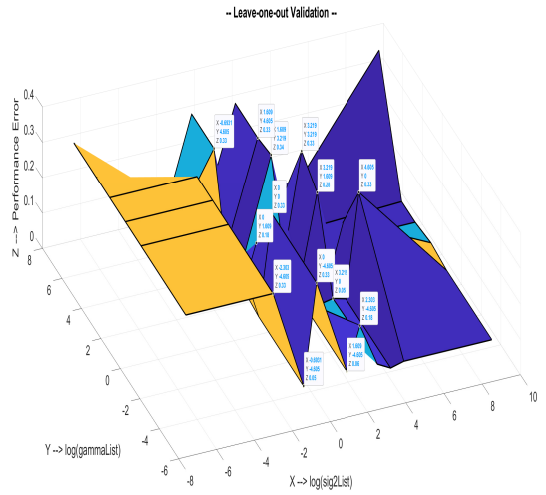
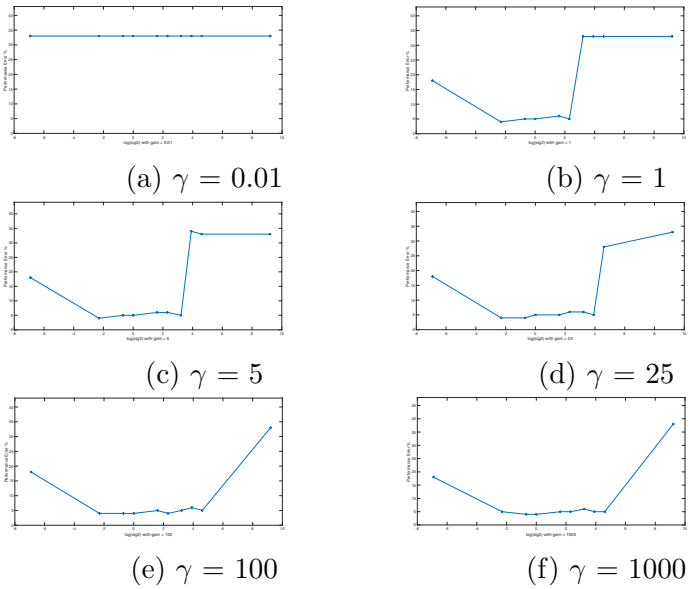


Figure 1.15: Leave-one-out Validation - ($\log(\sigma^2)$ vs log (γ) vs Performane Error)



(a) $\gamma = 0.01$ (b) $\gamma = 1$ (c) $\gamma = 5$ (d) $\gamma = 25$ (e) $\gamma = 100$ (f) $\gamma = 1000$

Figure 1.16: Iris data : LOOCV ($\log(\sigma^2)$ values for each level of γ Vs Performance Error) (%).

Why should one prefer crossvalidation over simple validation (random split)? How to choose the value of k in k-fold crossvalidation?

In a cross validation method the model is trained k times over k-1 subsets of data each time. This way the overall error of the model is averaged over the errors of each iteration. By doing this, the overall variance of the end result is small when compared with random split validation. This can be seen in figures 1.11 and 1.13. The value for k is chosen in a such way that the data samples in each set of k splits can statistically represent the original dataset. The choice of k influences the bias-variance trade-off. Factors like size of the data set, computational complexity, etc should also be taken into account while choosing k. For smaller data sets, a higher value of k is chosen so that the training sets have sufficient data samples. For larger data sets, a value of 10 can be chosen. In practice a value of 5 or 10 is chosen as optimal value as these values have proven to give unbiased results.

Automatic parameter tuning

Try out the different 'algorithm'. What differences do you observe? Why do the obtained hyperparameters differ a lot in different runs? What about the cost? Computational speed? Explain the results.

Simplex Algorithm			Grid Search Algorithm		
γ	σ^2	cost	γ	σ^2	cost
3.54	0.018991	0.04	18.165	0.026866	0.04
23.508	0.34958	0.04	133.44	0.024016	0.02
0.4226	0.18471	0.04	98.508	63.077	0.04
6.0573	0.19643	0.04	3944.3	0.32205	0.03
106.55	0.085258	0.03	0.2905	0.020474	0.03
12.893	0.21712	0.04	11.129	0.01182	0.03
382.45	0.28002	0.03	0.36295	0.23467	0.04
1.7285	2.6992	0.04	0.046299	0.41124	0.04

Table 1.5: Optimal (γ, σ^2) pairs using auto tuning methods

We train the SVM model again on Iris data set using tunelssvm for 8 iterations. The function trainlssvm performs 10-fold cross validation on training data and returns optimal values for γ and σ^2. In previous section of tuning hyper-parameters,

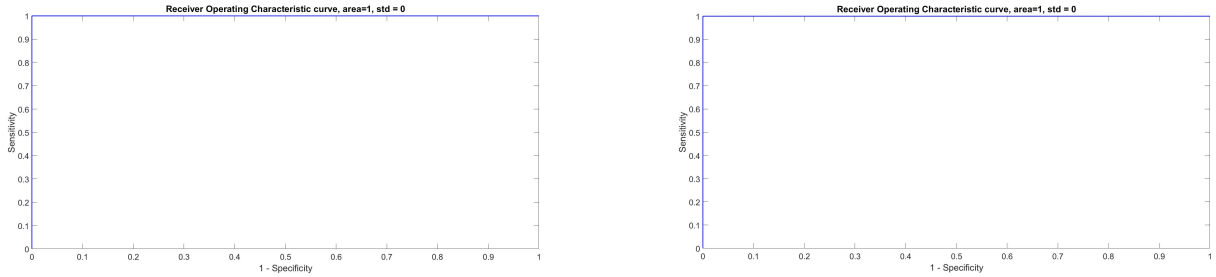
we saw that for a specific value of γ a range of σ works better (shown in table 1.4). How can we decide which pair of (γ, σ^2) is optimal. This is where the optimization algorithms simplex and grid search comes. They solve the convex-optimization problem for (γ,σ^2) and arrive at a local optima. We calculate the average time taken for each algorithm to give optimal values of (γ,σ^2).
Time taken to train SVM for 8 iterations using simplex algorithm is **3.058253 secs**.
Time taken to train SVM for 8 iterations using grid search algorithm is **6.016854 secs**.
We notice that simplex algorithm takes lesser time compared to grid search. This is because simplex tries to converge to the nearest local optima by initializing some random values to γ and σ^2 whereas grid search evaluates over a range of parameters and then looks for the local optima.
From table 1.5 it can be seen that the cost for all (γ,σ^2) pairs is around 0.02 to 0.04 which indicates that these pairs are optimal. As seen earlier in tuning hyper-parameters, the results show that many different (γ,σ^2) optimal pairs are possible. Therefore, the difference in (γ,σ^2) pairs for each run in auto tuning method is expected.

Using ROC Curves

In practice, we compute the ROC curve on the test set, rather than on the training set. Why?

ROC curves is one of the evaluation methods for the performance of a classifier. The classifier model is built using the training data. Evaluating the model for classification accuracy using training set will only give a biased estimate as the model has already seen the data samples. Therefore, the proper way to evaluate the classifier is using unseen data samples i.e., using test sets.

Generate the ROC curve for the iris.mat dataset (use tuned gam and sig2 values). Interpret the result.



(a) ROC Curve with (γ,σ^2)=(6.0573,0.19643) obtained from auto tuning of hyper-parameters using simplex algorithm

(b) ROC Curve with (γ,σ^2)=(18.165,0.026866) obtained from auto tuning of hyper-parameters using grid search algorithm

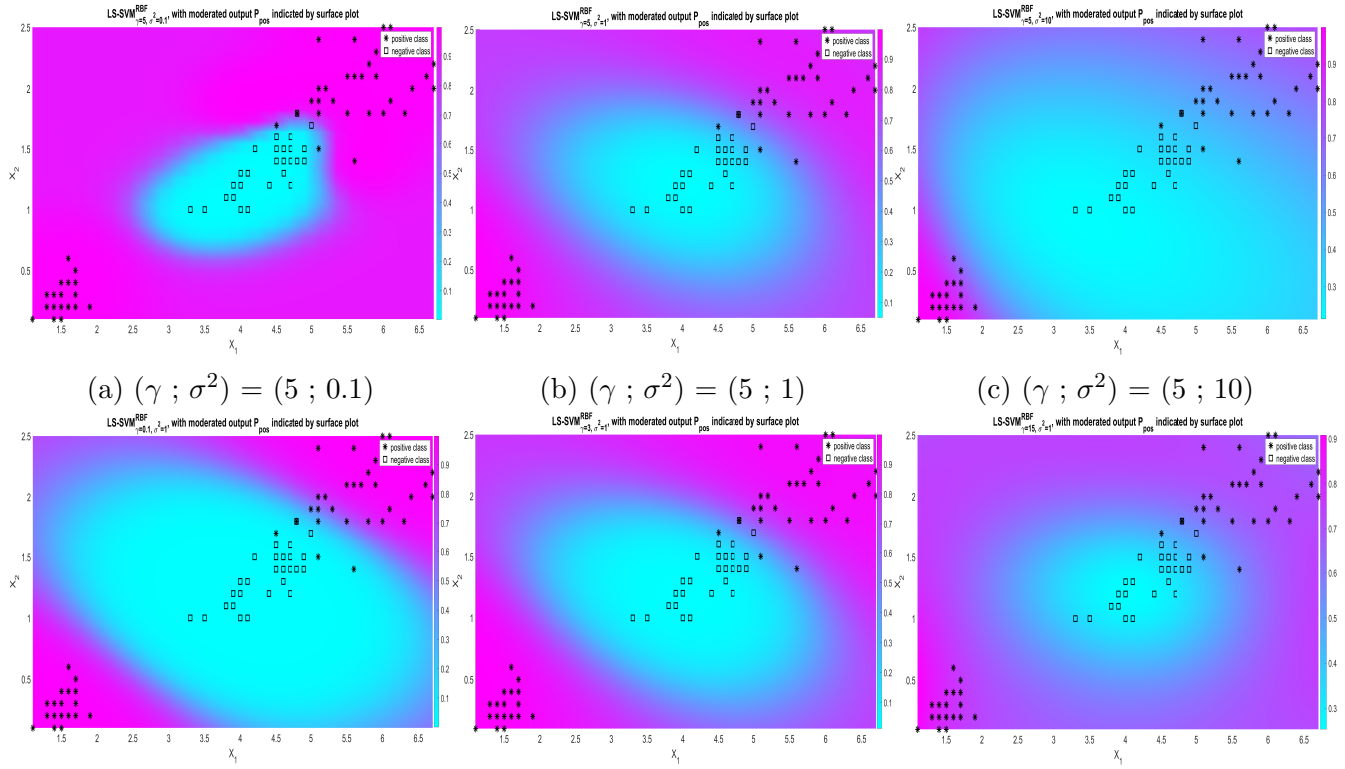
Figure 1.17: ROC Curves for SVM classifier with optimal (γ,σ^2) pairs

The ROC curves in the figure 1.17 tested on IRIS data set resembles an ideal classifier. Therefore, our SVM model built with optimal parameters of (γ,σ^2) pair is a perfect model.

Bayesian framework

How do you interpret the colors of the plot? Change the values of gam and sig2. Visualize and discuss the influence of the parameters on the figure

The Bayesian framework unlike other methods of SVM cannot give a strong decision boundary. Rather it gives a probability of a data sample belonging to a target class based on prior and posterior probabilities of the data distribution. From the figures in fig 1.18, it can be observed that the bayesian method gives two colors for each class (cyan - negative and magenta - positive) that slowly fade into one other indicating a separation of the two classes without a decision boundary. The colors here denote the probability of finding a positive class. As seen from the values of color bars next to each plot, it can be understood that magenta indicates the high probability of data points of positive class whereas cyan indicates the least probability of finding a positive class and vice versa. To get the results as show in figure 1.18, the bayes framework is run first with a fixed value of γ and varying value of σ^2 and then with a fixed value of σ^2 and varying value of γ.



(a) $(\gamma\ ;\ \sigma^2) = (5\ ;\ 0.1)$ (b) $(\gamma\ ;\ \sigma^2) = (5\ ;\ 1)$ (c) $(\gamma\ ;\ \sigma^2) = (5\ ;\ 10)$

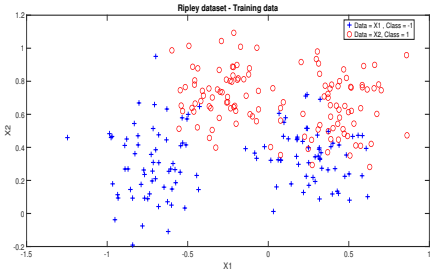
(d) $(\gamma\ ;\ \sigma^2) = (0.1\ ;\ 1)$ (e) $(\gamma\ ;\ \sigma^2) = (3\ ;\ 1)$ (f) $(\gamma\ ;\ \sigma^2) = (15\ ;\ 1)$

Figure 1.18: Bayesian Framework for various $(\gamma\ ;\ \sigma^2)$ values

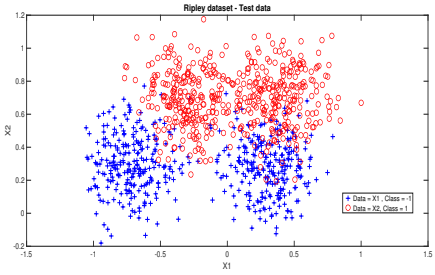
The plots 1.18a, 1.18b. 1.18c is obtained by fixing the γ value to 5 and varying the σ^2 values to (0.1; 1 ; 10). The plots 1.18d, 1.18e. 1.18f is obtained by fixing the σ^2 value to 1 and varying the γ values to (0.1; 3 ; 15). It can be seen from the plots that for a higher value of γ and lower value of σ^2, the model is able to accurately classify the data points of two classes. When σ^2 becomes greater than γ ($\sigma^2 \geq 2\gamma$) in fig 1.18c and 1.18d the cyan region becomes more dominant than magenta indicating a uncertainty in model. This leads to a lot of misclassifications. If the σ^2 values is made too small, the classification region can become too small trying to fit all the data points causing over fitting. Therefore, from these observations it can be said that it is preferred to have a balance between the values of $(\gamma\ ;\ \sigma^2)$ having a σ^2 value lesser than γ for better performance of the model.

1.2 Homework Problems

1.2.1 Ripley Dataset



(a) Training samples of ripley dataset

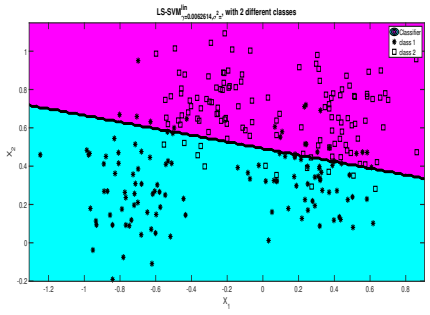


(b) Test samples of ripley dataset

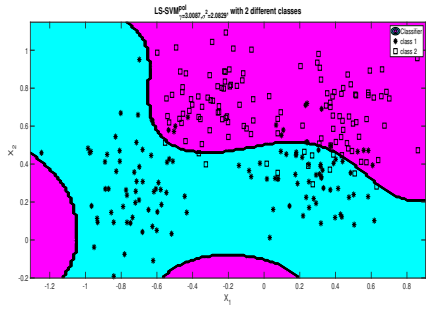
Figure 1.19: Visualizing training and test samples of ripley data

The data set contains 250 training samples and 1000 test samples with only two features. Therefore, it is easy to visualize on a 2D plot. From the plot 1.19 it can be seen that the data not completely linearly separable as there are some overlaps between the two classes. So, a non-linear decision boundary can separate these two classes better than a linear one. However, we train the SVM for linear , polynomial, RBF kernels and evaluate the models using ROC curves. The optimal hyper parameters of (γ, σ^2) for training is obtained using auto tuning method with simplex algorithm as seen in the earlier sections.

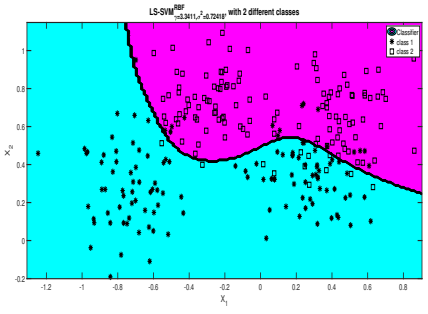
As seen from the figures 1.20b, 1.20c the classifier with polynomial and RBF kernel performs in a similar way allowing only few misclassifications whereas the classifier with linear kernel gives few more misclassifications. In addition to this, the classifier is trained for few iterations with different optimal hyper parameters generated by *tunelssvm* function and some of their results are evaluated and tabulated in 1.6. The results in table 1.6 and figures 1.21a, 1.21b, 1.21c show that, the classifier with RBF kernel gives slightly better accuracy (96.86 % from ROC curve) than polynomial or linear kernels.



(a) SVM with linear kernel- γ = 0.0062614



(b) SVM with pol kernel- γ = 3.0087 and degree = 5



(c) SVM with RBF kernel- (γ, σ^2) =(3.3411, 0.72418)

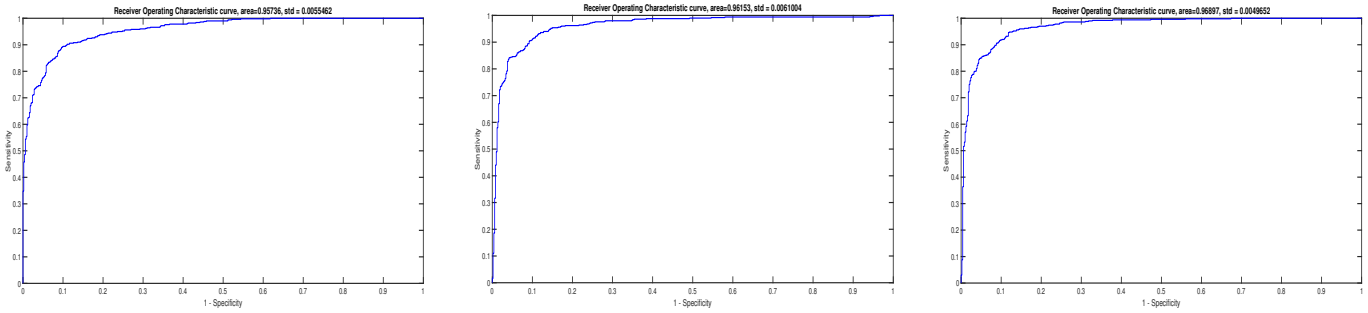
Figure 1.20: SVM Classifier trained on Ripley dataset

1.2.2 Wisconsin Breast Cancer dataset

This data set contains 400 training samples and 169 test samples. There are 30 features in this dataset and is difficult to visualize on either 2D or 3D plots. As seen from the ROC curves in figure 1.22, the SVM classifier with linear and RBF kernels gives an accuracy greater than 99%. This could be because of the reason that both the classes are well separated without many overlaps

Linear Kernel			Polynomial Kernel			RBF Kernel		
γ	σ^2	Error	γ	σ^2	Error	γ	σ^2	Error
0.00291	-	10.6	0.12909	1.109	10.2	0.59161	0.28572	9.3
0.01678	-	10.5	3.0087	2.0829	10.1	8.5381	0.60447	9.5

Table 1.6: Ripley dataset - Performance of SVM classifier with different kernels for auto tuned hyper parameters



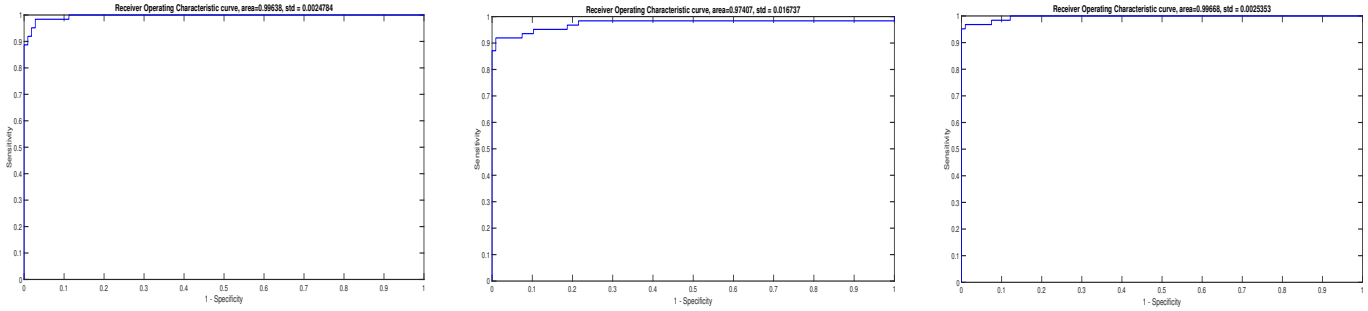
(a) ROC curve of SVM classifier with polynomial kernel

(b) ROC curve of SVM classifier with polynomial kernel

(c) ROC curve of SVM classifier with RBF kernel

Figure 1.21: ROC curves to evaluate SVM Classifier trained on Ripley dataset

and the model is able to classify the test samples to the right class. Similar to the previous ripley data, the model is trained for few iterations with different optimal hyper parameters generated by *tunelssvm* function and some of their results are evaluated and tabulated in 1.7. It is observed that the model performance remain more or less the same for change in hyper parameters in the case of linear and RBF kernels. For polynomial kernel, the model performs better for a degree of 3 than 5.



(a) ROC curve of SVM classifier with polynomial kernel

(b) ROC curve of SVM classifier with polynomial kernel

(c) ROC curve of SVM classifier with RBF kernel

Figure 1.22: ROC curves to evaluate SVM Classifier trained on breast cancer dataset

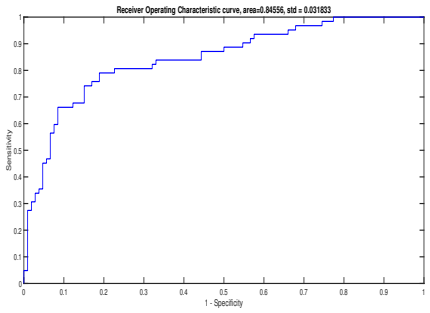
1.2.3 Diabetes dataset

This data set contains 300 training samples and 168 test samples. There are 8 features in this dataset and is difficult to visualize on either 2D or 3D plots. As seen from the ROC curves in figure 1.22, the SVM classifier with polynomial kernel(fig 1.23b) performs with lesser accuracy(81.4 %) compared with linear(fig 1.23a) and RBF kernels (1.23c) which gives an accuracy of 84%. To

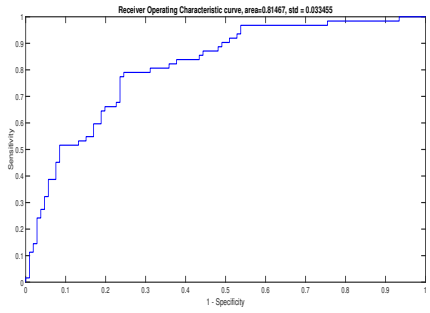
Linear Kernel			Polynomial Kernel			RBF Kernel		
γ	σ^2	Error	γ	degree	Error	γ	σ^2	Error
0.07253	-	4.7337	1.533e-05	3	4.7337	03.261	37.617	2.3669
8.2338	-	4.7337	2.0799e-07	3	3.5503	22.922	37.755	1.7751
0.47591	-	4.142	182.19	5	15.976	168.12	44.754	4.142

Table 1.7: Breast Cancer dataset - Performance of SVM classifier with different kernels for auto tuned hyper parameters

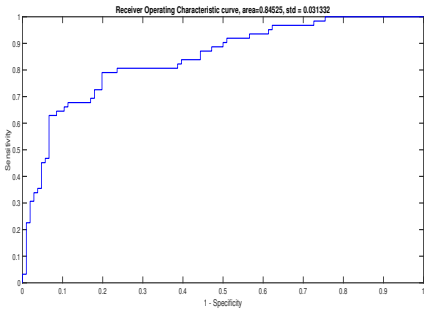
investigate this performance accuracy of the model, two of the features were plotted and it is observed that the two classes have over lapping samples as shown in figure 1.24. Therefore, the model finds it difficult to generate a good decision boundary that separates these two classes with greater accuracy. Similar to the two previous data sets, the model is trained for few iterations with different optimal hyper parameters generated by *tunelssvm* function and some of their results are evaluated and tabulated in 1.8. It is observed that the RBF kernel performs better and the model performances of all three models remain more or less the same despite the change in hyper parameters.



(a) ROC curve of SVM classifier with polynomial kernel



(b) ROC curve of SVM classifier with polynomial kernel



(c) ROC curve of SVM classifier with RBF kernel

Figure 1.23: ROC curves to evaluate SVM Classifier trained on diabetes dataset

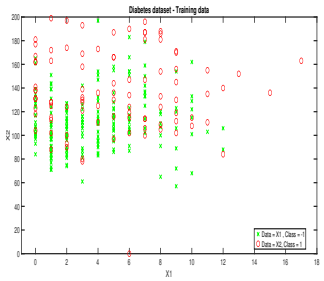


Figure 1.24: Diabetes dataset

Linear Kernel		Polynomial Kernel			RBF Kernel		
γ	Error	γ	degree	Error	γ	σ^2	Error
0.0136	25	0.0020	3	29.167	70.514	253.93	21.429
0.0176	23.214	1.2677	3	37.5	191.73	94556.	22.619
0.0090	25	0.0942	3	32.143	4.149	255.93	22.619

Table 1.8: Diabetes dataset - Performance of SVM classifier with different kernels for auto tuned hyper parameters

2. Exercise 2: Function Estimation and Time Series Prediction