FACULTY OF ENGINEERING

MASTER OF ARTIFICIAL INTELLIGENCE

ACADEMIC YEAR 2018-2019

Support Vector Machines: Methods and Applications

Exercise Sessions Report

Anthoula Mountzouri (r0736452)

Prof. Johan Suykens

Contents

Exercise session 1: Classification

1.1 A simple example: Two Gaussians

An artificial dataset is constructed from two classes of Gaussians with the same covariance matrices. Figure 1 shows a geometric construction used to estimate the optimal classifier between the two classes. As it is known from literature, for two data classes with equal covariance matrices, the decision boundary is linear, both in the case of a small overlap as well as in the case of a large overlap between distributions, meaning that this is in fact independent of how much the two classes are overlapping.

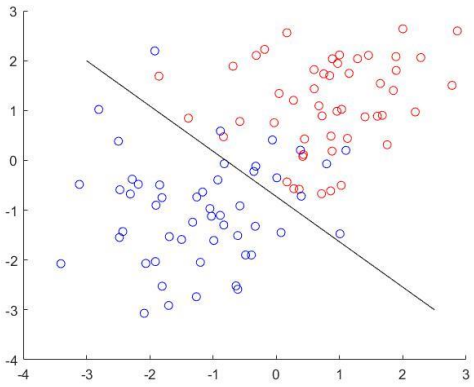


Figure 1 – Geometric Construction of Linear Classifier

In this particular example, a small overlap can be observed between the two classes and as a result, it is unavoidable that some misclassifications exist (we see that some blue points are assigned to the wrong class). Thus, the linear decision boundary may be valid only if one tolerates the misclassifications and the optimality condition corresponds to minimizing the probability of misclassification.

In more details, according to Bayesian decision theory, the rule for obtaining minimal probability of misclassification is to assign a pattern x to a class i* such that

$$i^* = \arg \max_{i=1,\ldots,n_{\mathcal{C}}} P(\mathcal{C}_i|x)$$

and hence maximize posterior class probability $P(C_i|x)$. In other words, if someone wants to classify a particular pattern x, could check the posterior probabilities $P(C_i|x)$ for all of the classes and then could assign the pattern x to that class which is giving the maximal posterior class probability. Like that, if someone does decision making in this way, he/she is going to minimize the probability for the misclassification. This is illustrated in Figure 2 for a binary classification problem.

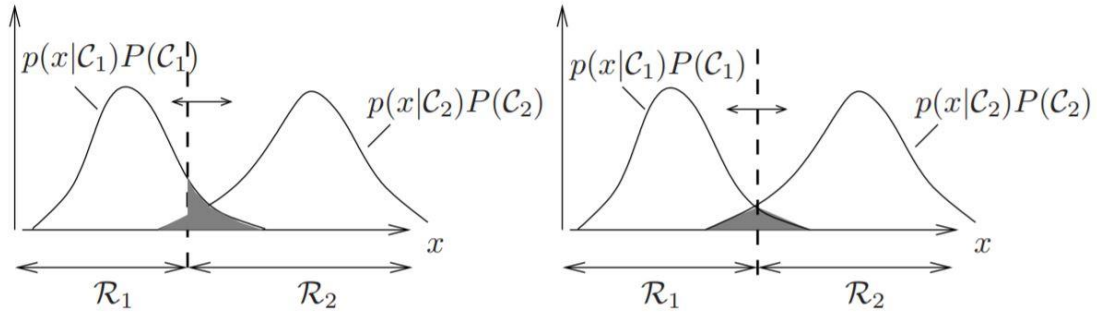


Figure 2 – A minimal shaded area corresponds to minimizing the probability of misclassification and maximizing the posterior probability.

In the Figure 2, we see the curves for two classes C_1 and C_2, and if the decision threshold is at the dashed line, then the dashed area is indicative for the misclassification that we make (the probability of misclassification). So, we can minimize that shaded area by taking the vertical line right at the intersection of the two curves. We, see that the shaded area in the graph at the right end side, is smaller comparing to the shaded area in the graph at the left end side. And that is because the decision threshold is going right through the intersection point of the two curves. Now, if we do decision making at the point of the vertical line, by taking that line at that position, that corresponds to doing the decision making by looking at the maximum posterior class probability. So, the same applies for our example, where calculating the probability distribution of X1 and X2 and by taking the vertical line right at the intersection of the two curves, we can obtain the minimal probability of misclassification.

1.2 Support vector machine classifier

A Support Vector Machine (SVM) is a Machine learning Algorithm, that can be employed for both classification and regression purposes. SVMs are commonly used in classification problems and the objective of such an algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.

If we consider two input variables x_1 and x_2 and some labeled training data points (it's a supervised learning problem) of two classes, we can see that there are many possible lines that could be chosen to separate the two classes, as illustrated in Figure 3.

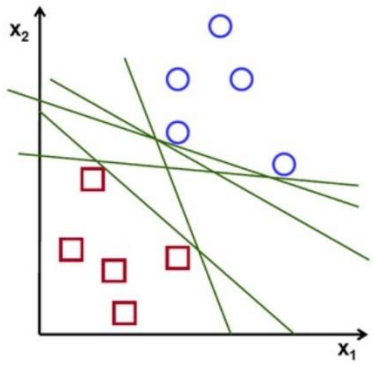


Figure 3 – Illustrates some possible hyperplanes that could be used to separate the two classes

In the context of SVM, though, the aim would be to come up with a unique line-classifier that is perfectly separating the two classes from each other and in order to do that we focus on the red cubes of Class1 and the blue circle of Class2, that is to say that points of Class1 that are as close as possible to the points of Class2 (Figure 4). These points are called **support vectors** and influence the position and the orientation of the hyperplane, since they are closer to this hyperplane, and are crucial to come up with a formulation of a margin of a support vector machine. So, the support vectors are part of the training data. That is to say, we are looking for that separating line, that classifier otherwise, that is going to maximize the margin, and by using these support vectors, we maximize this margin of the classifier. Deleting the support vectors will change the position of the hyperplane. This idea is illustrated in the following figure.

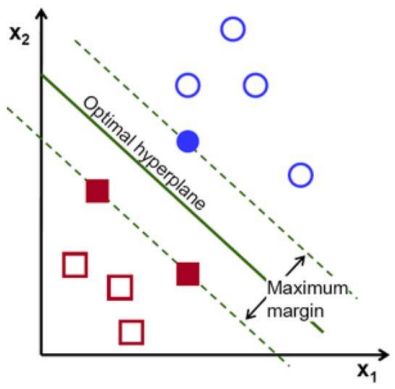


Figure 4 – Illustrates the optimal hyperplane that separates the two classes

So, in the general case of a linear classifier, the decision boundary between the two classes will correspond to the equation $w^T x + b = 0$, where w is a parameter vector and b the bias term, and the expression for a classifier is in fact $\hat{y} = sign[w^T x + b]$, which will be the estimation of the class label (either +1 or -1). The margin or in other words the distance between the (dashed) lines that are formed by the support vectors, is equal to $\frac{2}{||w||_2}$. Minimizing the term $||w||_2$ corresponds to maximizing the margin and as a result the optimal classifier can be achieved. So, it is actually a constrained optimization problem, taking also into account the constraint that all the training data points must be correctly classified.

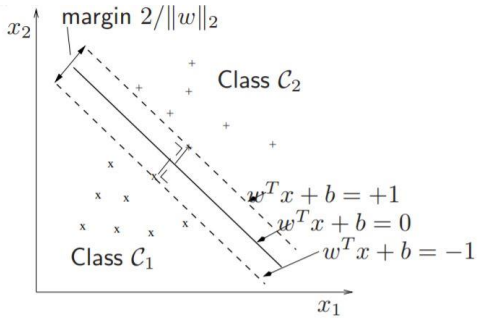


Figure 5 - Illustrates the decision boundary

So, it turns out that in the case of having a linear SVM classifier and separable data, the primal optimization problem is

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{s.t.} \quad y_k[w^T x_k + b] \geq 1, \quad k = 1, ..., N$$

and it turns out that the primal representation of the classifier is $\hat{y} = sign[w^T x + b]$ while the dual representation turns out to be $\hat{y} = sign[\sum_{i=1}^{N} a_i y_i \boldsymbol{x_i}^T \boldsymbol{x} + b]$, as it is formed according to the Lagrangian Multipliers α_i, that are related to the constraints. The Lagrangian Multipliers will be as many as the training data points but due to the sparsity property, it turns out that most of them will be zero and only the Lagrangian multipliers that correspond to the support vectors will be non-zero.

In the case of having training data that are non-separable, we have to tolerate misclassifications. For that reason, the so-called slack variables are introduced in the objective function. The objective function has then two terms; the first is going to maximize the margin and is related to the regularization term and the second term $c\sum_{k=1}^{N} \xi_k$, which is involving the slack variables ξ, is tolerating the amount of misclassifications. So, the tuning parameter 'c' is a kind of trade-off choice and the value of it depends on how much someone wants to emphasize the first term with respect to the second term. So, it is going to express how much emphasis we put on maximizing the margin versus tolerating misclassifications in our formulation. It is not something that follows from the convex problem, it is something that we select, but the value of it can prove to be of crucial importance and can affect the whole result of the classification.

Optimization problem (primal problem):

$$\min_{w,b,\xi} \mathcal{J}(w,\xi) = \frac{1}{2} w^T w + c \sum_{k=1}^{N} \xi_k$$

subject to

$$\begin{cases} y_k[w^T x_k + b] \geq 1 - \xi_k, & k = 1, ..., N \\ \xi_k \geq 0, & k = 1, ..., N. \end{cases}$$

However, in most of the cases the datasets are probably never linearly separable, so the condition of being classified correctly by a hyperplane will never be met. So, SVM addresses non-linearly separable cases by introducing the concept of Kernel Tricks.

In more details, using a function ϕ() which is called feature map, the training points from the input space are mapped to a high dimensional feature space, as it is called, and then instead of doing the classification in the original input space, we are doing a linear classification in the high dimensional feature space. It is like transforming the input data to a new space, the feature space, and then do linear classification in this new space. So, in that case the optimization problem gets the following format

Optimization problem (**non-separable case**):

$$\min_{w,b,\xi} \mathcal{J}(w,\xi) = \frac{1}{2} w^T w + c \sum_{k=1}^{N} \xi_k$$

subject to

$$\begin{cases} y_k[w^T \varphi(x_k) + b] \geq 1 - \xi_k, & k = 1, ..., N \\ \xi_k \geq 0, & k = 1, ..., N. \end{cases}$$

As a result, the classifier in its dual representation is $\hat{y} = sign[\sum_{i=1}^{N} a_i y_i \, \boldsymbol{\varphi}(\boldsymbol{x_i})^T \boldsymbol{\varphi}(\boldsymbol{x}) + b]$, where what we see now is that the inner product $\boldsymbol{x_i^T x}$ between these data points from the input space is replaced by the product $\boldsymbol{\phi(x_i)^T \varphi(x)}$, since the initial points were mapped to the high dimensional feature space. According to Mercer theorem, if a positive definite Kernel function exists, the product $\boldsymbol{\phi(x_i)^T \varphi(x)}$ will be equal to that kernel function $K(x_k, x_l)$ and in that case we don't need the knowledge of this feature map anymore. The classifier in its dual representation is characterized by the following format, then:

$$y(x) = \text{sign}[\sum_{k=1}^{N} \alpha_k \, y_k \, K(x, x_k) + b]$$

There are many choices for this kernel function. Some possible kernels are the linear and the RBF kernel and some examples concerning these two kernels are presented below. On the website https://cs.stanford.edu/people/karpathy/svmjs/demo/, we are given the chance to experiment with support vector machine (SVM) classification and these two kernels, using the appropriate online application that is hosted there. Some data points are added to the initial dataset, and while experimenting between different values of the tuning parameters, the following results emerge:

Liner Kernel:

Actually, it is the case of having a linear SVM and it corresponds to a kernel function that is just taking the inner product of the data points in the input space. This linear kernel has the following format:

$$K(x, x_k) = x_k^T x \text{ (linear SVM)}$$

Tuning Parameter c:

In the case of the linear kernel, the only tuning parameter that affects the result of the classification is the regularization constant 'c' that as is mentioned above, it acts as a trade-off choice, expressing how much emphasis some puts on maximizing the margin versus tolerating misclassifications in the formulation. So, if the value of **'c' is very large**, this means that the optimization will choose a **smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly**. Therefore, we will not allow almost any misclassification, which might lead to overfitting. On the other hand, if the value of the tuning **parameter 'c' is small**, this means that there is **more emphasis on maximizing the margin**, even if the hyperplane that occurs misclassifies more points. For **tiny** values of the parameter c, someone should get **misclassified examples often**, even if the training data are linearly separable.

Subsequently, we try out different values of the regularization hyperparameter c using the given online application and provide the results below.

- ***C is very small (tiny)-0.016***:
  When launching the application for the first time, someone can notice that the margin is large (Figure 6).

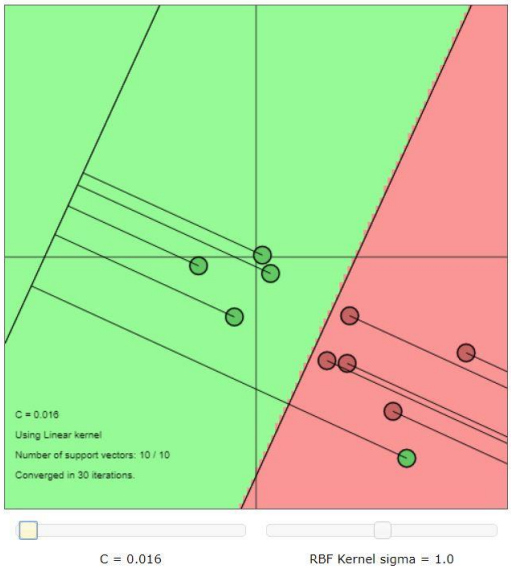


Figure 6 - C =0.016 - Initial Dataset

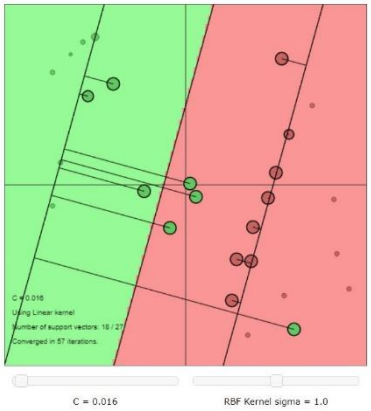


Figure 7 - Adding data on right side

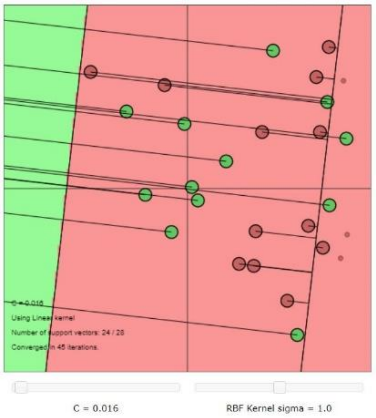


Figure 8 - Adding data on wrong side

By adding more data, many misclassifications occur even in the case of having linearly separable data (Figure 7). In the case of adding the additional data on the wrong side, the margin is getting even bigger (Fig. 8).

- ***C is small - 1:***

  Launching the data for the first time, the following classification occurs:

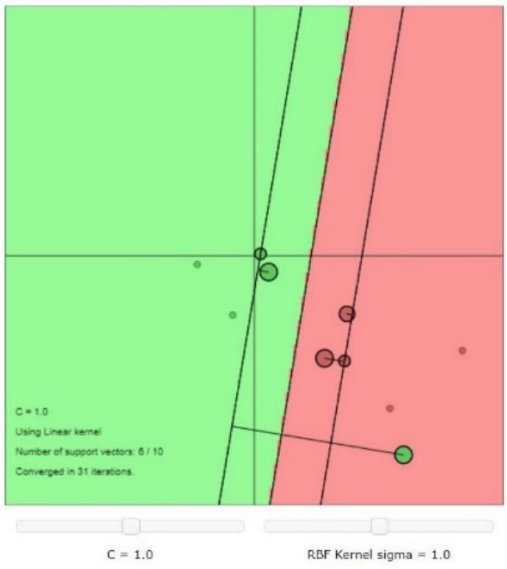


Figure 9 - C=1.0 - Initial Dataset

When the tuning parameter c=1.0, in general we observe that adding more data on the right side, changes each time the position and the orientation of the hyperplane, so that take into consideration the new data and find a proper decision boundary, but it is not

affecting much the size of the margin. However, on the other hand, when adding data on the wrong side, the margin becomes very large, the position and orientation of the decision boundary changes again and many misclassifications occur. The number of support vectors also increases (almost all the data points are made support vectors).

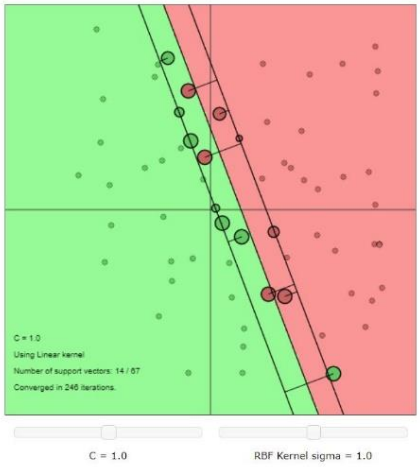


Figure 10 - C=1, Adding data on right side

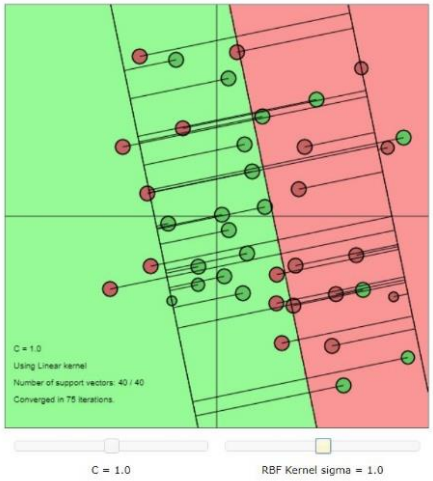


Figure 11 - C=1, Adding data on wrong side

- ***C is large:***

Launching the data for the first time, we get the classification presented in the next figure. The margin in this case, tends to be very small, much smaller than those presented in previous cases.

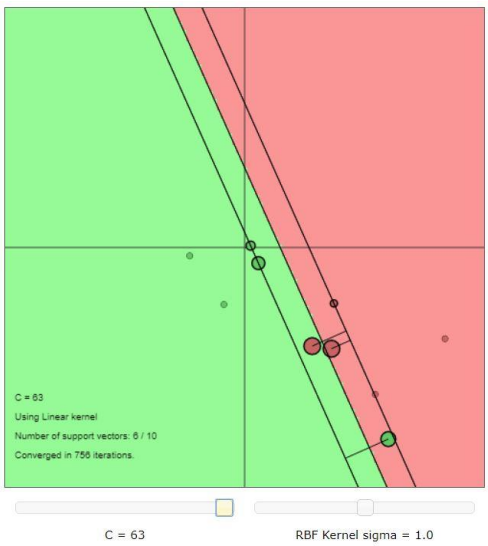


Figure 12 – C=63, Initial Dataset

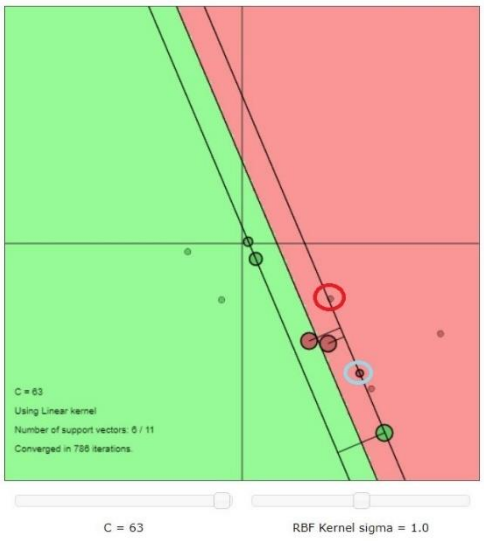


Figure 13 – C=63, Close to decision boundary

Adding more data on the right side, the margin becomes smaller and smaller, such that the data are classified better (Figure 14).

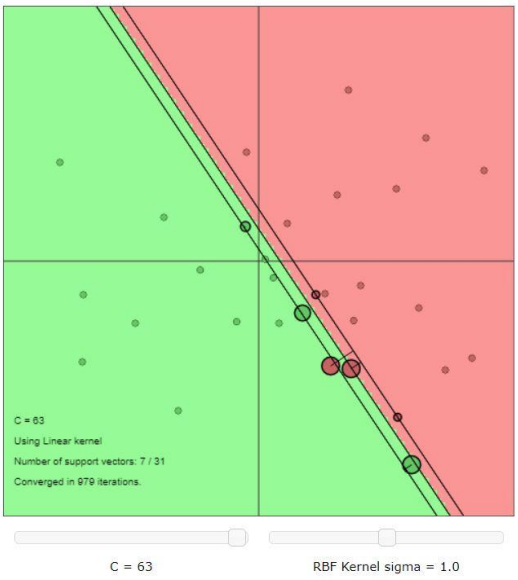


Figure 14 – C=63 - Adding Data on right Side

Even in the case of adding data on the wrong side, the margin becomes smaller, so that acquiring better classification (Fig.15), and avoid the appearance of misclassifications, apart from the case, however, of adding data far enough from the decision boundary, where the margin becomes bigger trying to classify the data correctly.

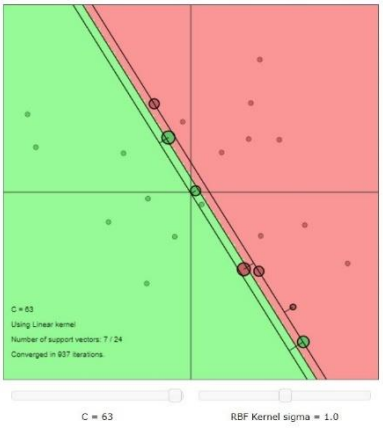


Figure 15- C=63, Adding Data on wrong side

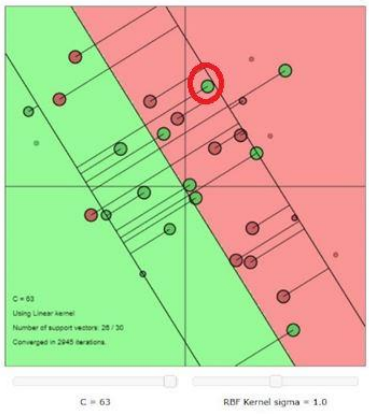


Figure 16- Adding Data on wrong side - away from decision boundary

What we can understand during the experiments presented, is that a particular data point becomes a support vector if it is close enough to the decision boundary and contributes to

the definition of the margin during the optimization process, after which the data point would have a corresponding $\alpha_k \neq 0$. The importance of the support vector is related to the value of its corresponding α_k (support value); the higher the value, the greater the importance of the support vector. In Figure 16, we can see that the new point (pointed out with a red circle) becomes a support vector in this case, because it is located on the wrong side of the decision boundary representing a misclassification. Therefore, generating that the slack variable for this data point gets penalized, while having a greater influence in the minimization of the objective function. Also, in Figure 13, is visually illustrated in red, the case of a data point that previously was a support vector and due to the new data point added which is pointed out with a light blue circle, it is not a support vector anymore, while the new data point becomes a support vector. So, as the new data points make the decision boundary shift, other data points that are in the surrounding area either become a support vector or lose this property.

RBF Kernel:

It is probably the most popular case and has the following format.

$$K(x, x_k) = \exp(-\|x - x_k\|_2^2/\sigma^2) \text{ (RBF SVM)}$$

Here a Gaussian function exists, and it contains the term 'σ', which is the bandwidth of the kernel function. In this case, a very large 'σ' value means that the decision boundary tends to become linear, and small 'σ' value means that the decision boundary becomes highly nonlinear. However, of course the whole non-linear model is characterized by a single tuning parameter 'σ', at the dual level, but there is also the 'c' parameter in the following equation of the objective function that was already presented, and which has also to be tuned:

$$\min_{w,b,\xi} \mathcal{J}(w, \xi) = \frac{1}{2} w^T w + c \sum_{k=1}^{N} \xi_k$$

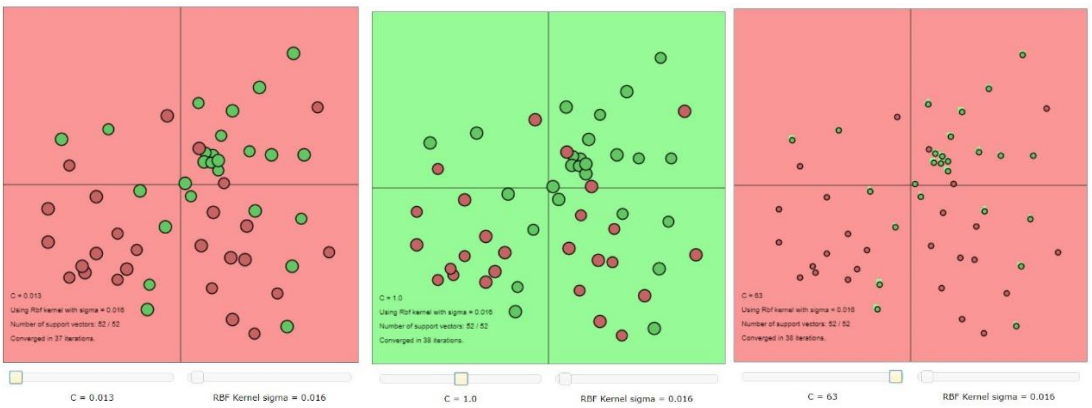
So, in this case, two tuning parameters exist. So, it is a convex optimization problem up to these two tuning parameters and it's very important to tune these parameters very well. Often, someone should try several combinations of these values and different scales and the best way of doing it, is to define a grid of different combinations of values and often on a logarithmic scale, because often the order of magnitude of these values is important. So, a grid with axes 'σ' and 'c' could be created, and in this way a grid with several combinations

of values occurs, and then a third axis could be added too, which could contain for example the error on a validation set. And like that, someone could check for which combinations he/she reached the minimum error on the validation set (trying many combinations c/σ, train the problem for different combinations and then check on an independent validation set, what the performance is and then pick out that combinations that give the smallest error on the validation set). These results can also be quite sensitive to the specific choice of the validation set. So, therefore, often 10-fold cross validation is considered, where the role of the validation part is changed by considering different runs, 10-runs in the case of 10-fold cross validation.

So, trying to see how this works in action, in this case of the RBF kernel, we try out different values of the regularization hyperparameter c and the kernel parameter σ, using the given online application and subsequently we provide the results.

- ***σ is 0.016 (tiny):***

  In this case, we observe that when the value of σ is too small, the Gaussian of the kernel is very piked, which generates that the region of coverage of each support vector is very small. In combination with a large value of the parameter c, we observe small islands of the green region surrounding each green support vector (Fig. 17(c)), as well as that the size of the support vectors is quite small. In the other two cases, where the parameter c is small (Fig. 17(a), Fig. 17(b)), the red or the green class respectively, is assigned to all the training data points, which means that the points are not classified properly.

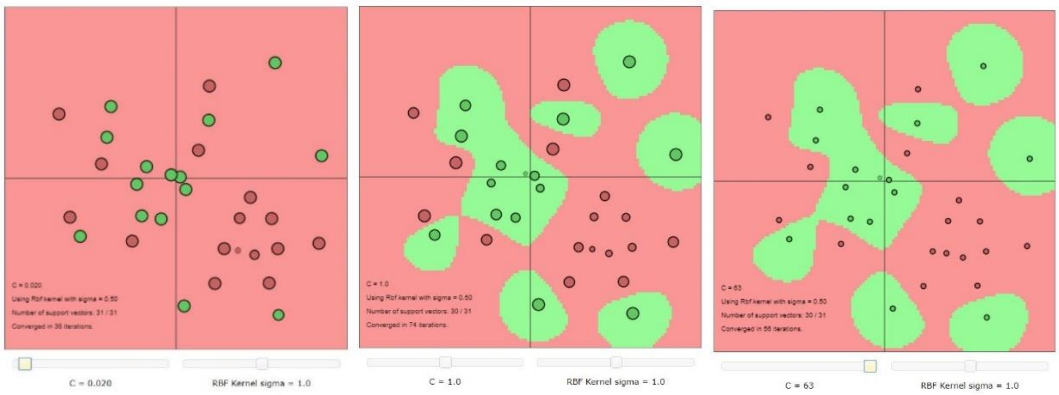


(a) C=0.013 **(b) C=1.0** **(c) C=63**

Figure 17 - Effect of variation of the kernel hyperparameter σ=0.016. RBF kernel

- ***σ is 1:***

In this scenario, we observe that the classification fits the underlying distribution, which has a circular shape (the decision boundary becomes totally non-linear) and separates properly the data points belonging to the two classes. An exception is observed in the first case (Fig.18 (a)) in which misclassifications occur, with the red class covering all the training data points. Also, it is worth mentioning that the bounds of the two classes are a bit sharp and that the size of support vectors in the third case (Fig. 18 (c)) is quite small.

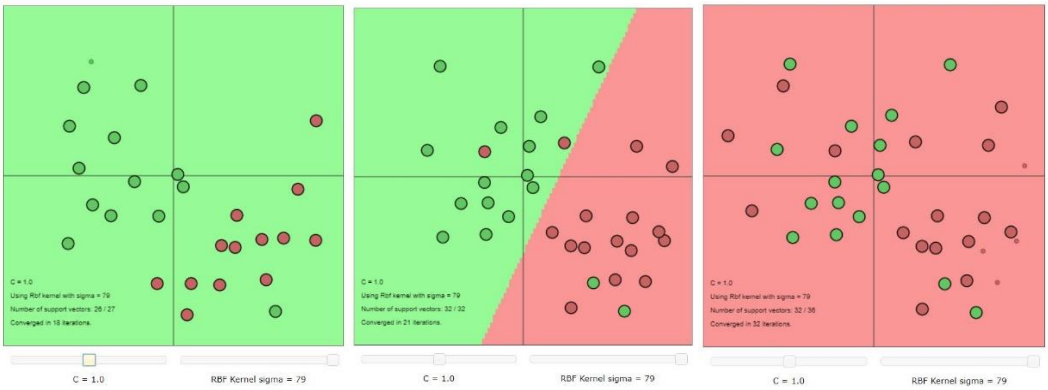


(a) C=0.020 **(b) C=1.0** **(c) C=63**

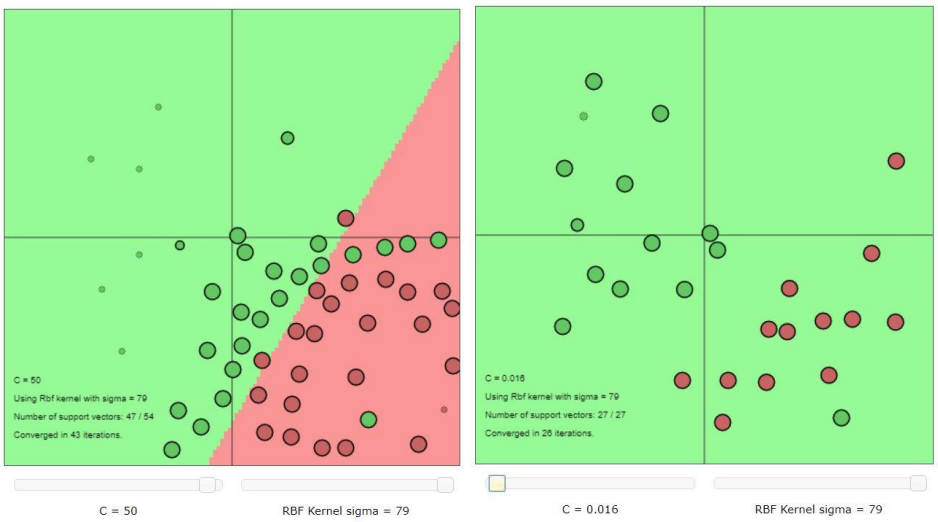
Figure 18 - Effect of variation of the kernel hyperparameter σ=1. RBF kernel

- ***σ is 79 (large):***

Here, someone can observe that the decision boundary tends to become linear, no matter what values are assigned to the regularization parameter 'c'. One can also observe that in some cases such as Fig. 19(a), Fig. 19(c), Fig. 19(e) (no matter when we insert points to the right or wrong side) the red or the green class respectively, is assigned to all the training data points, which is translated to misclassifications.



(a) C=1.0 (b) C=1.0 (c) C=1.0

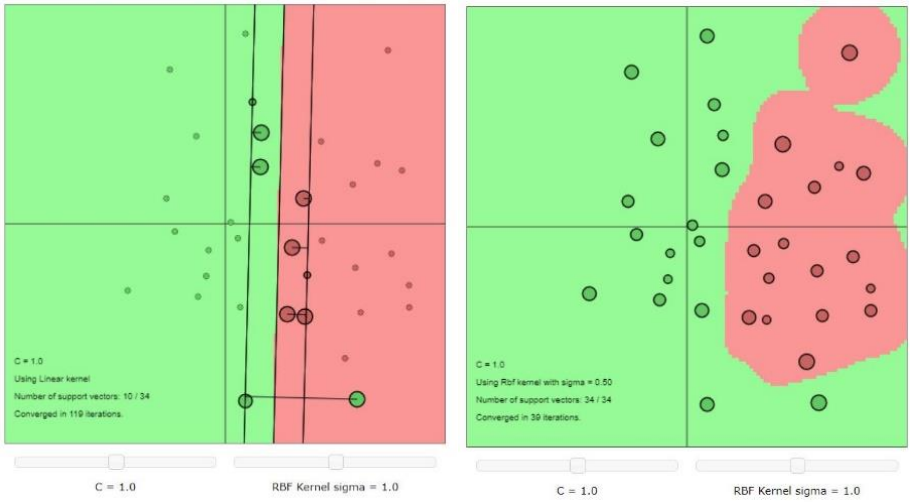


(d) C=50 (e) C=0.016

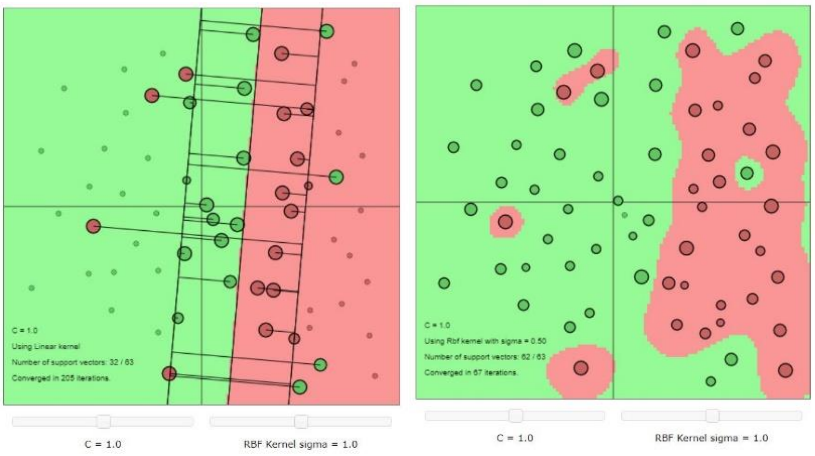
Figure 19 - Effect of variation of the kernel hyperparameter σ=79. RBF kernel

Compare Classification using linear and RBF kernel:

Subsequently, we compare the performance of an SVM classifier using linear kernel with that of a classifier using an RBF kernel. For this reason, both a linear and a non-linear task are used. For the linear task, we can observe that both the SVM with the linear kernel (Fig.20 (a)) and with the RFB kernel (Fig.20 (b)) perform very well and are able to deal with outliers, giving a correct classification and generalization. However, for the linear task, the SVM with the linear kernel obtains a mores sparse solution (less amount of support vectors). On the other hand, for the non-linear task, the RBF kernel can easily fit the circular shape of the underlying distribution (Fig.20 (d)), while the SVM classifier with the linear kernel cannot fit the non-linear underlying distribution (Fig.20 (c)). In this case, the classifier with the RBF kernel converges faster but the SVM with the linear kernel obtains a mores sparse solution. As a result, we could conclude that with a proper selection of hyperparameters the SVM with an RBF kernel is probably going to be able to perform as good or even better than a classifier using a linear kernel, either for a linear or for a non-linear task. However, if we are interested in an optimal solution, which is fast and sparse, too, we should choose the right kernel for the specific problem.



(a) Linear Kernel (c=1, σ=1) **(b) RBF Kernel (c=1, σ=1)**



(c)Linear Kernel (c=1, σ=1) **(d) RBF Kernel (c=1, σ=1)**

Figure 20 - SVM with linear kernel vs. RBF kernel, for a linear (top) and non-linear (bottom) classification task.

1.3 Least-squares support vector machine classifier

Subsequently, the least squares based variant of the support vector (LS-SVM) is considered, using the the LS-SVMlab toolbox (https://www.esat.kuleuven.be/sista/lssvmlab/). The experiments in this part are restricted to the classification of Iris data set.

1.3.1 Influence of hyperparameters and kernel parameters

Polynomial Kernel

Initially, a polynomial kernel with degree=1,2,3,4,5 and t=1 (fix γ=1) is used. Performance is evaluated on the test set. The results of the experiments demonstrate a significant improvement in the performance of the LS-SVM model for polynomial kernels of higher degree. In the following figures, the LS-SVM results are plotted in the environment of the training data.

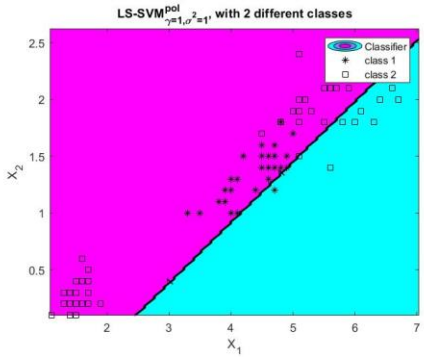


Figure 21- Polynomial kernel with degree=1

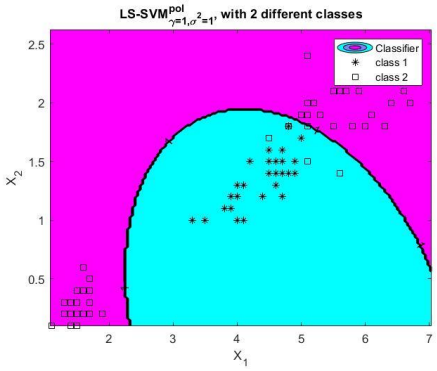


Figure 22-Polynomial kernel with degree=2

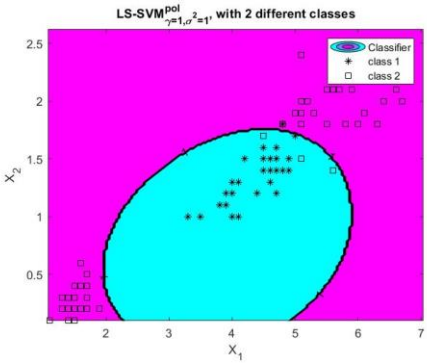


Figure 23-Polynomial kernel with degree=3

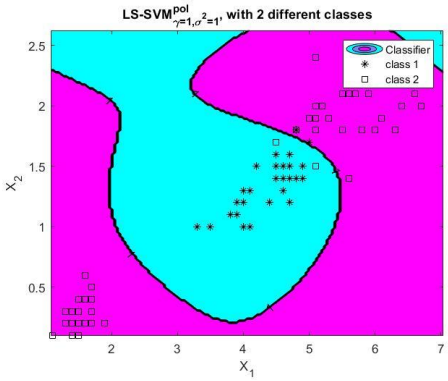


Figure 24-Polynomial kernel with degree=4

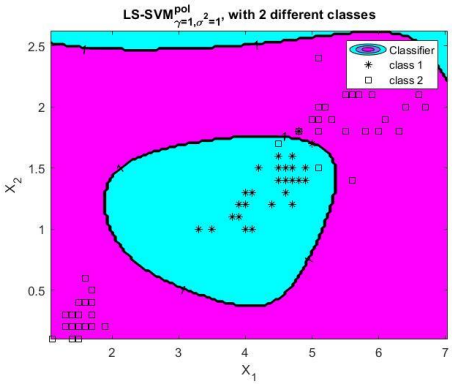


Figure 25-Polynomial kernel with degree=5

Performance of the polynomial kernel on the test set:

- *Degree = 1: Misclassified data on test set = 11/20, Error Rate on test set= 55.00%*
- *Degree = 2: Misclassified data on test set = 1/20, Error Rate on test set= 5.00%*
- *Degree = 3: Misclassified data on test set = 0/20, Error Rate on test set= 0.00%*
- *Degree = 4: Misclassified data on test set = 0/20, Error Rate on test set= 0.00%*
- *Degree = 5: Misclassified data on test set = 0/20, Error Rate on test set= 0.00%*

It may be observed that by increasing the degree of the polynomial kernel the performance of the LS-SVM model improves significantly.

Polynomial Kernel, t=1

Degree	Error Rate
1	55%
2	5%
3	0%
4	0%
5	0%

Table 1: Performance of the polynomial kernel on iris data set

RBF Kernel

An RBF kernel is considered now, with squared kernel bandwidth σ^2. Initially, the performance of the LS-SVM model is examined in the classification task of a test set, on a **range of different σ^2** values as kernel parameters **(0.01, 0.1, 1, 5, 10, 25, 100)** and **fix γ=1**. In the following figures, the LS-SVM results are plotted in the environment of the training data.

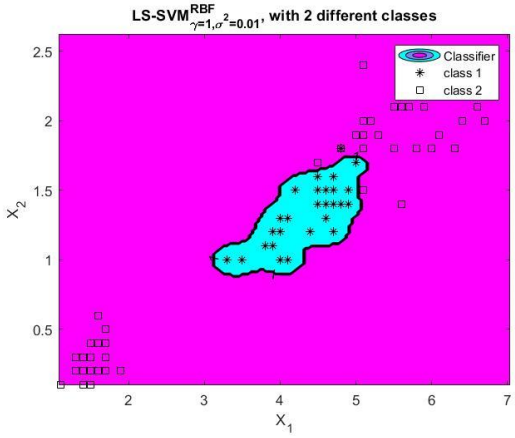


Figure 26- RBF kernel with sig2=0.01

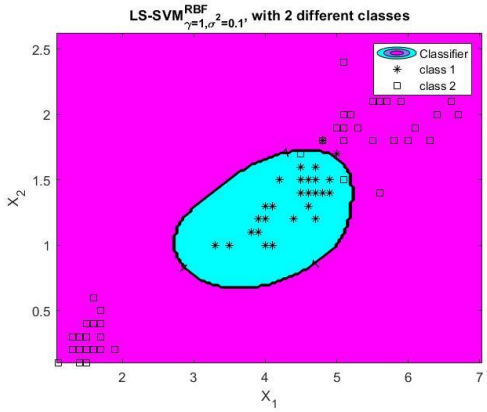


Figure 27- RBF kernel with sig2=0.1

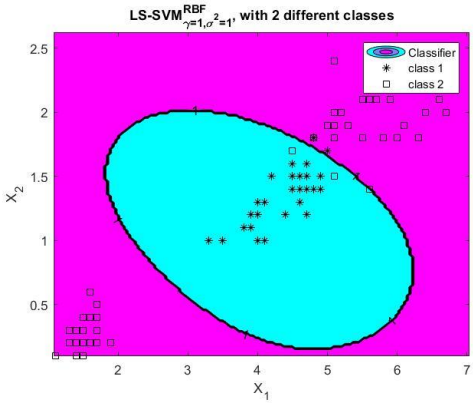


Figure 28- RBF kernel with sig2=1

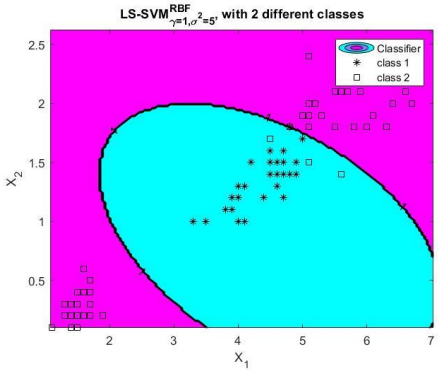


Figure 29- RBF Kernel with sig2=5

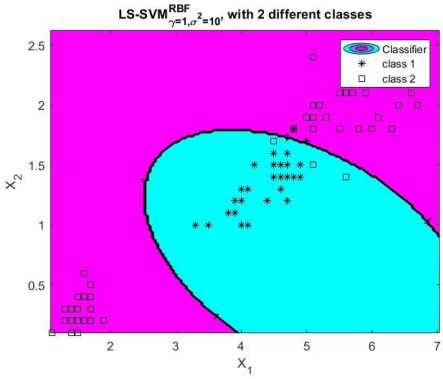


Figure 30- RBF Kernel with sig2=10

Performance of the RBF kernel on the test set (tune σ^2, fix γ):

- ***σ^2= 0.01:*** *Misclassified data on test set = 2/20, Error Rate on test set= 10.0%*
- ***σ^2= 0.1****: Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- ***σ^2= 1:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- ***σ^2= 5:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- ***σ^2= 10:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- ***σ^2= 25:*** *Misclassified data on test set = 10/20, Error Rate on test set= 50.0%*
- ***σ^2= 100:*** *Misclassified data on test set = 10/20, Error Rate on test set= 50.0%*

One may observe that the performance is optimal for σ^2 values equal to 0.1, 1, 5 and 10 whereas for values smaller than 0.1 and larger than 10 misclassification errors occur.

RBF Kernel, γ=1	
σ^2	Error Rate
0.01	10%
0.1	0%
1	0%
5	0%
10	0%
25	50%
100	50%

Table 2 - Performance of the RBF kernel, fix gam=1 and tune sig2 on iris data set

Also, in the following figure the misclassification rate on the test set, with respect to σ^2 is illustrated.

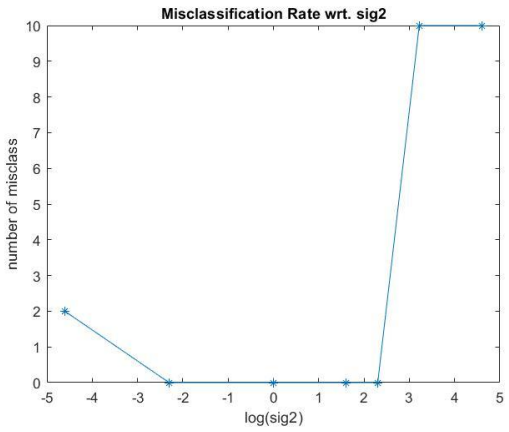


Figure 31 - Illustrates a range of sig2 with their corresponding test set performance

Afterwards, the performance of the LS-SVM model is examined in the classification task of a test set, on a **range of different gamma** values **(0.01, 0.1, 1, 5, 10, 25, 100)** and fix **σ^2=0.1** in this case. In the following figures, the LS-SVM results are plotted in the environment of the training data.

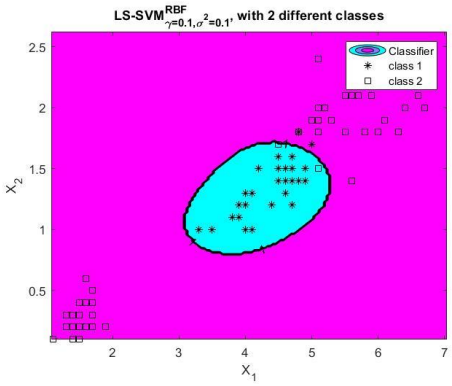


Figure 32 - RBF kernel with γ=0.1

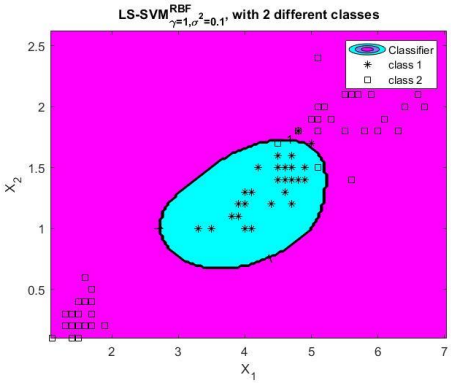


Figure 33 - RBF kernel with γ=1

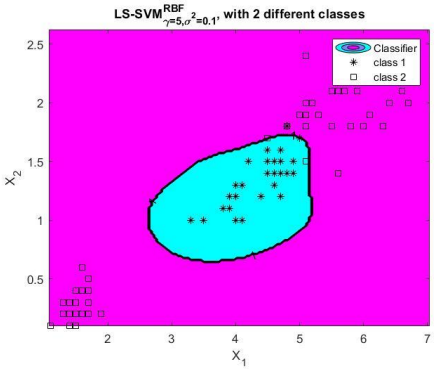


Figure 34 - RBF kernel with γ=5

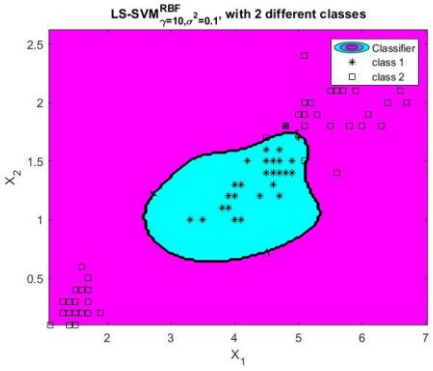


Figure 35 - RBF kernel with γ=10

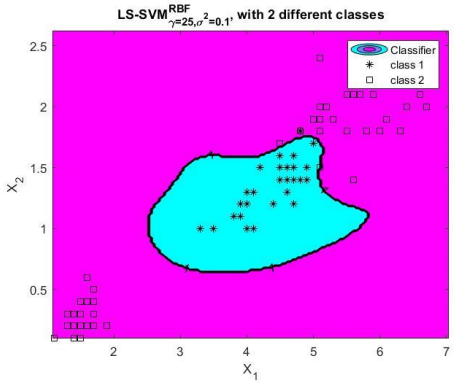


Figure 36 - RBF kernel with γ=25

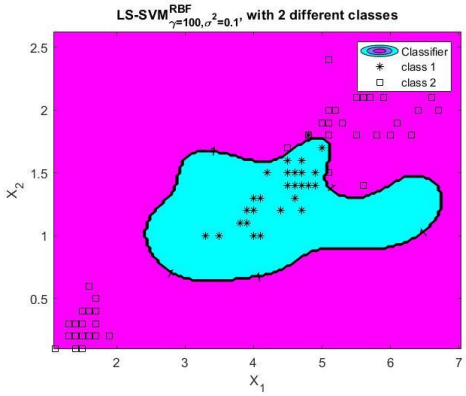


Figure 37 - RBF kernel with γ=100

Performance of the RBF kernel on the test set (tune gamma, fix $\sigma^2 = 0.1$):

- **γ= *0.01:*** *Misclassified data on test set = 10/20, Error Rate on test set= 50.0%*
- **γ= *0.1*:** *Misclassified data on test set = 2/20, Error Rate on test set= 10.0%*
- **γ= *1:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- **γ= *5:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- **γ= *10:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- **γ= *25:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- **γ= *100:*** *Misclassified data on test set = 1/20, Error Rate on test set= 5.0%*

One may observe that for γ values larger than 1 and approximately lower than 100, there are no misclassification errors whereas for values smaller than 1 and higher than 100, the performance drops and a number of misclassification errors occur. Therefore, a good choice of γ is important in order to achieve the desired performance.

RBF Kernel, σ^2=0.1

γ	Error Rate
0.01	50%
0.1	10%
1	0%
5	0%
10	0%
25	0%
100	5%

Table 3 Performance of the RBF kernel, fix sig2=0.1 and tune gamma on iris data set

In the following figure the misclassification rate on the test set, with respect to γ is illustrated (with fix $\sigma^2 = 0.01$).

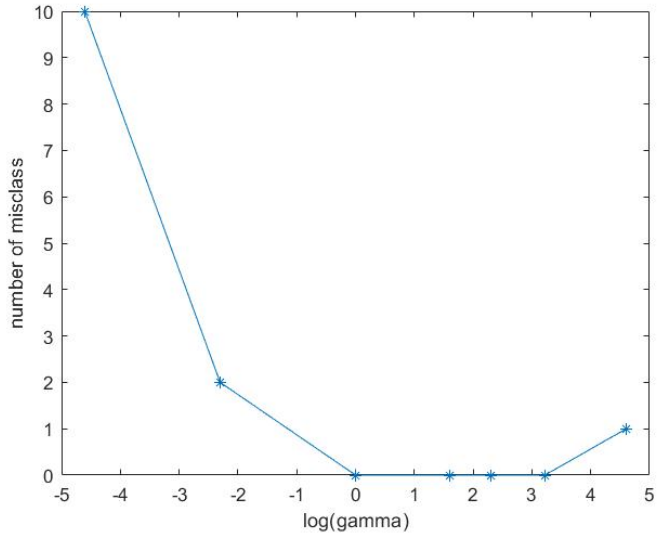


Figure 38 - Illustrates a range of gamma with their corresponding test set performance

The same experiments are repeated for the case of having a **range of different γ** values **(0.01, 0.1, 1, 5, 10, 25, 100)** and fix **σ^2=5.** The results are presented below.

<u>Performance of the RBF kernel on the test set (tune gamma, fix $\sigma^2 = 5$):</u>

- **γ= *0.01:*** *Misclassified data on test set = 10/20, Error Rate on test set= 50.0%*
- **γ= *0.1*** *: Misclassified data on test set = 10/20, Error Rate on test set= 50.0%*
- **γ= *1:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- **γ= *5:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- **γ= *10:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- **γ= *25:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*
- **γ= *100:*** *Misclassified data on test set = 0/20, Error Rate on test set= 0.0%*

RBF Kernel, σ^2= 5	
γ	Error Rate
0.01	50%
0.1	50%
1	0%
5	0%
10	0%
25	0%
100	0%

Table 4 Performance of the RBF kernel, fix sig2=5 and tune gamma on iris data set

In the following figure the misclassification rate on the test set, with respect to γ is illustrated (with fix σ^2 = 5 in this case).

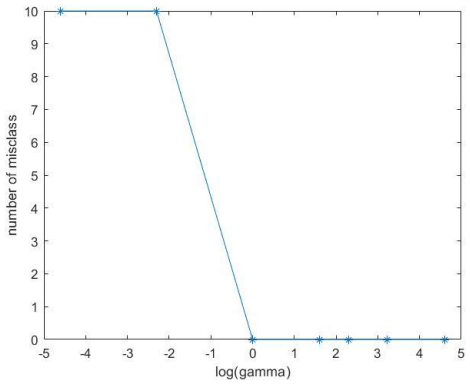


Figure 39 - Illustrates a range of γ with their corresponding test set performance

Observing now the new results, one can see that that for γ values larger than 1 there are no misclassification errors, whereas for values smaller than 1 the performance drops faster even in case of γ=0.1, where previously, for σ^2=0.01 the results were much better, and a number of misclassifications occur. As a result, someone could conclude that a good choice of γ is important in order to achieve the desired performance, but combining also the right value of σ^2 parameter, may affect also the result of the classification.

SampleScript_iris.m

Finally, in the file **SampleScript_iris.m**, the LS-SVM classifier is tested initially using a linear kernel of γ=1, then using a polynomial kernel of t=1 and degree=5 and lastly an RBF kernel of γ=1 and a range of different σ^2 values. The relevant results are presented below.

Linear Kernel, γ=1

on test: #misclass	Error Rate
11	55%

Table 5 - Performance of the linear kernel on iris data set (Results from file SampleScript_iris.m)

Polynomial Kernel, t=1, degree=5

on test: #misclass	Error Rate
0	0%

Table 6- Performance of the polynomial kernel on iris data set (Results from file SampleScript_iris.m)

RBF Kernel, γ=1

σ^2	on test: #misclass	Error Rate
0.01	2	10%
0.1	0	0%
1	0	0%
5	0	0%
10	0	0%
25	10	50%

Table 7 - Performance of the RBF kernel on iris data set (Results from file SampleScript_iris.m)

The misclassification rate on the test set with respect to σ^2, is illustrated (with fix γ= 1) in the following figure. Comparing the results of the file **SampleScript_iris.m** with those occurring from our experiments, it is obvious that they are the same.

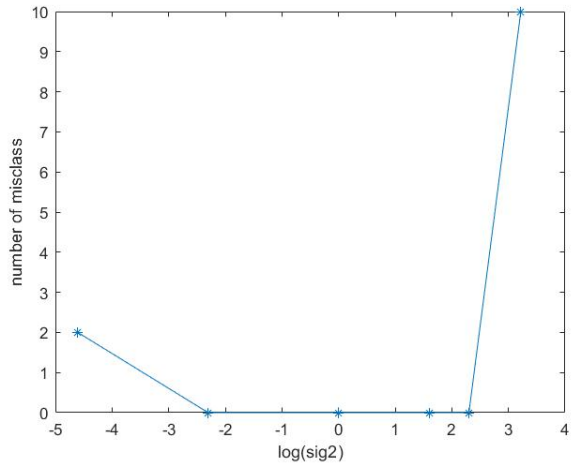


Figure 40 - Illustrates a range of sig2 with their corresponding test set performance

1.3.2 Tuning parameters using validation

Subsequently, some tuning algorithms are performed that split up the training data into a tuning and a validation part.

In more details, firstly, a random split is performed, which is one way of splitting up the training dataset into a training and a validation part by randomly taking some data points for training. The remaining data points are consequently used for validation (for example 80% of the training data is used for training, thus the 20% of the training data is used for validation).

Later, 10-fold cross validation is performed. There is evidence that one should prefer using this method over simple validation (random split). When data is randomly partitioned into the training and validation sets, there is some risk of creating a validation set that is missing some characteristics which are present in the full data set. This could cause an inaccurate estimate of the model's ability to generalize to unseen data. One way to reduce this risk is to repeat the train/validation split several times, each time calculating the error on the validation set. However, even with several splits, random partitioning does not guarantee that every point will appear at least ones in a test set. In k-fold cross validation, the original sample is randomly partitioned into K subsamples. Of the K subsamples, a single is retained as the validation data for testing the model, and the remaining K-1 subsamples are used as training data. The cross-validation process is then repeated K times (the folds) with each of the K subsamples used exactly once as the validation data. The K results from the folds then can be averaged (or otherwise combined) to produce a single estimation. The advantage of

this method over repeated random subsampling is that all observations are used for both training and validation, and each observation is used for validation exactly once. Table 8 illustrates the technique of cross-validation in an intuitive way. The validation set (=omitted part from the training) changes in each run.

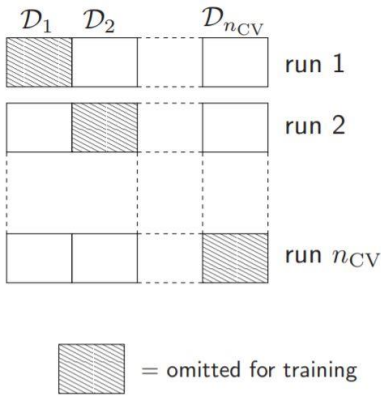


Table 8 - Illustrates the technique of cross validation

Finally, leave-one-out cross-validation involves using a single observation from the original sample as the validation data, and the remaining observations as the training data. This is repeated such that each observation in the sample is used once as the validation data. This is a special case of k-fold cross-validation, where k is taken equal as the number of data points. Leave-one-out cross-validation is computationally expensive because it requires many repetitions of training. So, that kind of technique is often used when you have only small data sets available.

The cost estimation of applying these performance estimators on Iris training dataset, is recorded below. Namely, the random split method, 10-fold cross validation and leave-one-out are used for a range of **γ** and **σ²** values (e.g. γ, $\sigma^2 = 10^{-3}$, ..., 10^3) and the results are described below (for the visualization part, the MATLAB function surf() was used).

Random Split:

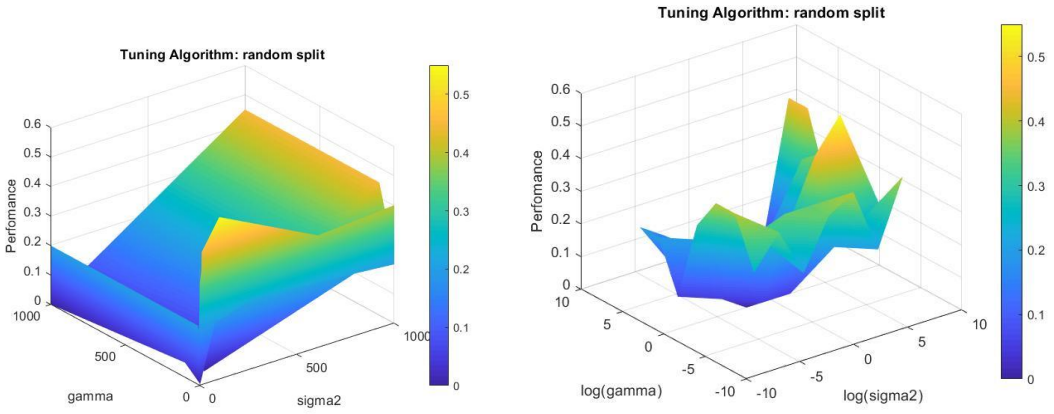


Figure 41 (a) Grid for random split error wtr. γ and σ² (b) Grid for random Split error wtr. log(γ) and log(σ²)

	$\sigma^2=10^{-3}$	$\sigma^2=10^{-2}$	$\sigma^2=10^{-1}$	$\sigma^2=1$	$\sigma^2=10$	$\sigma^2=10^{2}$	$\sigma^2=10^{3}$
$\gamma=10^{-3}$	10%	35%	30%	30%	35%	50%	40%
$\gamma=10^{-2}$	40%	35%	25%	35%	35%	25%	20%
$\gamma=10^{-1}$	40%	25%	10%	**0%**	30%	35%	35%
$\gamma=1$	25%	15%	5%	5%	5%	40%	25%
$\gamma=10$	35%	5%	10%	5%	**0%**	35%	20%
$\gamma=10^{2}$	10%	**0%**	5%	**0%**	10%	5%	55%
$\gamma=10^{3}$	20%	10%	5%	5%	15%	**0%**	40%

Table 9 - Random Split method error with variations in γ and σ² of RBF kernel on Iris dataset

However, it is worth mentioning that if someone tries to compute the performance of random split method more than once, different results are recorded each time and it makes sense, since each time the validation and the training set are selected randomly.

10-fold cross Validation:

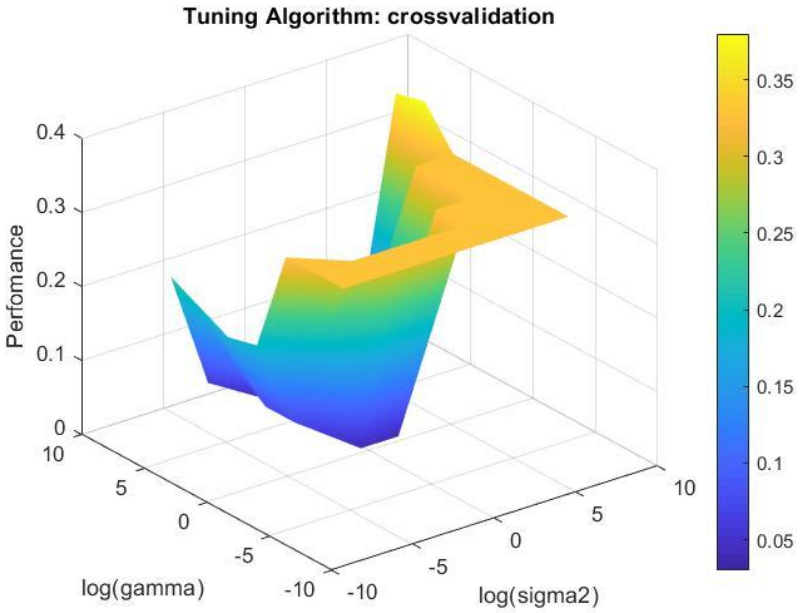


Figure 42 Grid for 10-fold cross validation error with respect to γ and σ^2

	$\sigma^2=10^{-3}$	$\sigma^2=10^{-2}$	$\sigma^2=10^{-1}$	$\sigma^2=1$	$\sigma^2=10$	$\sigma^2=10^2$	$\sigma^2=10^3$
$\gamma=10^{-3}$	33%	33%	33%	33%	33%	33%	33%
$\gamma=10^{-2}$	33%	33%	33%	33%	33%	33%	33%
$\gamma=10^{-1}$	33%	19%	**4%**	**5%**	33%	33%	33%
$\gamma=1$	20%	**7%**	**4%**	**5%**	**5%**	33%	33%
$\gamma=10$	20%	**5%**	**4%**	**5%**	**6%**	33%	33%
$\gamma=10^2$	20%	**5%**	**5%**	**4%**	**5%**	**3%**	37%
$\gamma=10^3$	18%	**4%**	**4%**	**4%**	**5%**	**5%**	37%

Table 10- 10-fold cross validation error with variations in γ and σ^2 of RBF kernel on Iris dataset

Leave-one-out Validation:

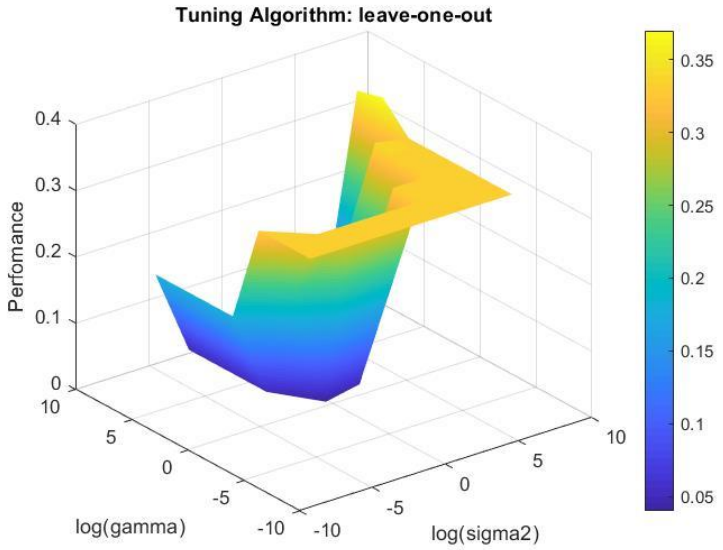


Figure 43 Grid for leave-one-out validation error with respect to γ and σ^2

	$\sigma^2=10^{-3}$	$\sigma^2=10^{-2}$	$\sigma^2=10^{-1}$	$\sigma^2=1$	$\sigma^2=10$	$\sigma^2=10^2$	$\sigma^2=10^3$
$\gamma=10^{-3}$	33%	33%	33%	33%	33%	33%	33%
$\gamma=10^{-2}$	33%	33%	33%	33%	33%	33%	33%
$\gamma=10^{-1}$	33%	20%	**4%**	**5%**	33%	33%	33%
$\gamma=1$	18%	**5%**	**4%**	**5%**	**5%**	33%	33%
$\gamma=10$	18%	**5%**	**4%**	**5%**	**6%**	34%	33%
$\gamma=10^2$	18%	**5%**	**4%**	**4%**	**4%**	**5%**	37%
$\gamma=10^3$	18%	**5%**	**5%**	**4%**	**5%**	**5%**	36%

Figure 11- leave-one-out validation error with variations in γ and σ^2 of RBF kernel on Iris dataset

Taking into consideration the results achieved from all those methods, one can observe that the optimal solution for the combination of hyperparameters is not unique, but instead there is huge variety of combinations that lead to a low validation error value.

Looking at the details, regarding initially the random spilt method, the information that someone can obtain about how good each combination of hyperparameter is, is actually very limited. Because of the randomness aspect of this method, it could happen that the selected validation set is not a representative sample of the training data. Then, the information that someone gets about the performance of the SVM when using a certain combination of hyperparameter is going to be biased towards a not very informative sample. As a consequent, using this technique, someone might wrongly choose a combination of hyperparameters for which got a very low validation error, but that was actually in a region from the search space where the error was in most of the cases a greater value.

On the other hand, 10-fold cross validation allows us to get a more informative result about the performance of each combination of hyperparameters. The same applies for the leave-one-out validation method, which is suitable only for small datasets though. Judging from the previous results, one can see that these two methods, achieve almost the same performance; the performance of leave-one-out validation method is slightly improved, and this might be, due to the fact that the available iris dataset is relatively small. So, according to the previous results, the combination of hyperparameters that generate the lower misclassification error is in the intersection region between: $1 \leq \gamma \leq 10^3$ and $10^{-2} \leq \sigma^2 \leq 10$.

1.3.3 Automatic parameter tuning

In this part, the optimization of the hyper-parameters is performed automatically with the Matlab function tunelssvm(). The optimal values of the hyper-parameters may one view in the following tables. One may observe by running the tunelssvm command several times that there exist several sets of optimal hype-parameters values.

Simplex Algorithm:

γ	σ^2
8.5375	0.017873
10.4699	0.392995
0.38456	0.10625
0.044014	0.81781
0.041196	0.47154

Table 12 - Set of optimal hyper-parameters tuned with the method of cross validation using the simplex algorithm

Grid Search Approach:

γ	σ2
0.22161	0.37748
0.045034	0.49525
0.46291	0.23859
0.19935	0.54367
0.22616	0.29353

Table 13 - Set of optimal hyper-parameters tuned with the method of cross validation using a grid search approach

The two optimization algorithms that are used in this tunelssvm() function is either the brute force grid search (gridsearch) or the Nelder-Mead method (simplex). Grid search is a two-dimensional optimization method based on exhaustive search on a limited range. It is the simplest algorithm to determine the minimum of a cost function with possibly multiple optima by taking a grid over the parameter space in order to pick the minimum. This procedure iteratively zooms to the candidate optimum. Simplex is an alternative optimization function which also finds local minima in a multidimensional space. Suitable starting points for every method are determined by Coupled Simulated Annealing (CSA) and these are passed to the simplex method in order to fine tune the result. Both of them find good results but probably not the optimal solution. Furthermore, one may observe that grid search approximation function is slower. Also, the fact that the hyperparameters differ a lot in different runs is because of the stochastic nature of CSA.

1.3.4 Using ROC curves

In this part the receiver operating curve (ROC) is considered on the Iris data set. The ROC curve gives information regarding the quality of the classifier. The higher the area under the curve the better the classifier separates data. However, considering the ROC curve as a performance measure, a test set of unseen data is recommended, instead of a training set, in order to obtain a good representation of the model's ability to classify and have a good

generalization. Otherwise, if the ROC performance is indicated on the training set it may be a case of overfitting.

Figure 44 shows the ROC for the iris.mat dataset (using tuned γ and σ^2 values). As the area under the curve is equal to 1, it means that the classifier obtains a perfect result for the test data with which it is evaluated. This shows that there is no overlap between the classes on the test set, therefore being separable and allowing the classifier to get a perfect classification result.

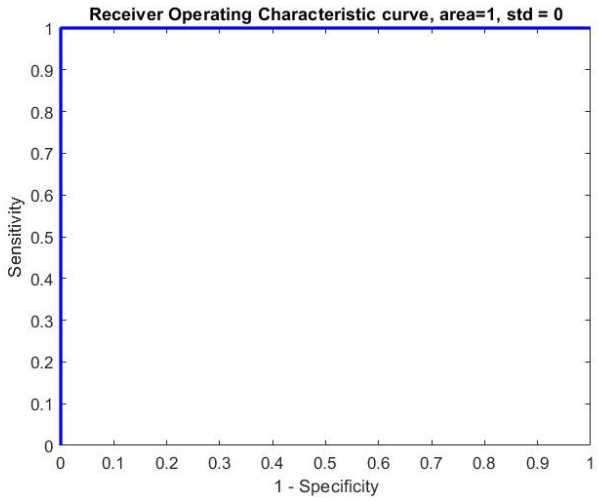


Figure 44 -ROC curve evaluated on Iris test set.

1.3.5 Bayesian framework

A Bayesian framework is now used to get probability estimates. The iris data set is considered again in this case. The next figures illustrate the influence of changing the values of γ and σ^2 in the frame of Bayesian inference. The Bayesian inference is a probabilistic approach that provides us with confidence levels indicated by a range of colors (colormap). The colors of the plot can be interpreted by using the color bar of the figure; the probability of belonging to the positive class (*) is greater for the purple color and lower for blue color. No binary output exists anymore, instead we have probability distribution over how likely is for a given region of the input space to belong to the positive class. So, as it can be observed in the next figures, for small values of the bandwidth σ, the decision boundary is sharper, giving more certainty in the classification, whereas for larger values the boundary becomes smoother introducing uncertainty levels. The shape of the classification also differs for different values of γ.

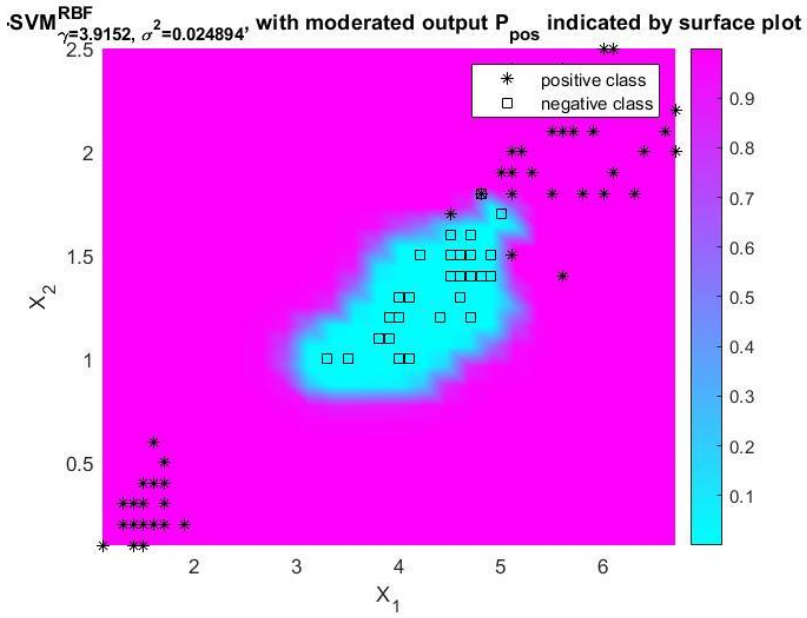


Figure 45 - γ= 3.9152, σ2 = 0.024894

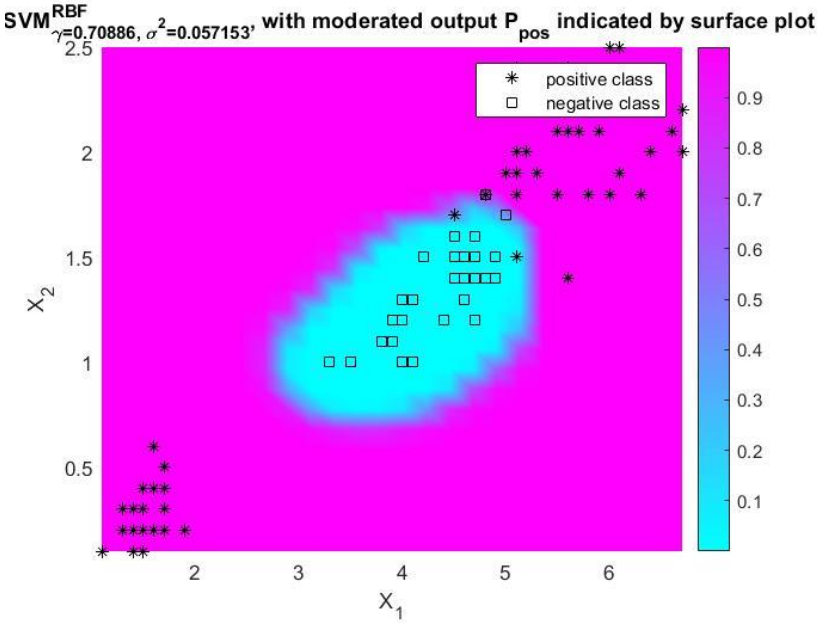


Figure 46 γ=0.70886, σ²=0.057153

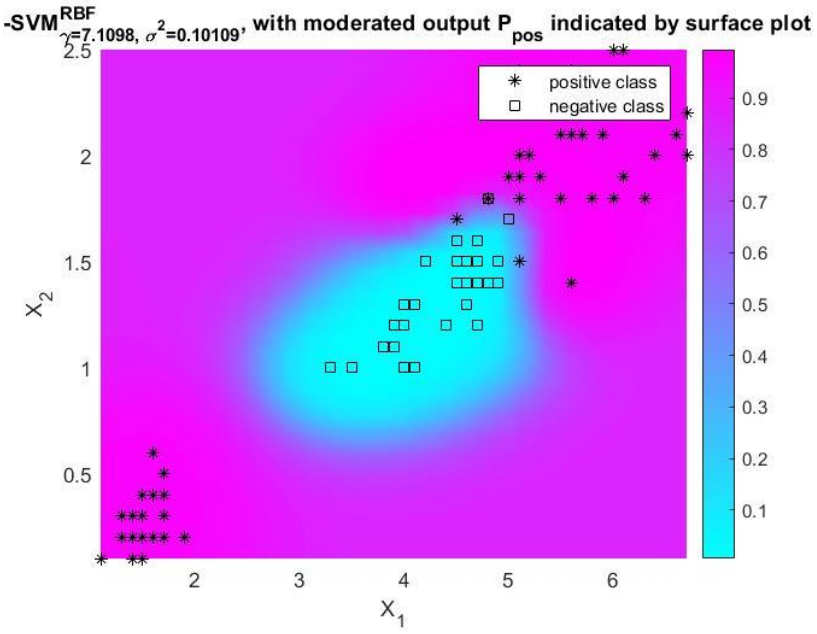


Figure 47 γ=7.1098, σ2=0.10109

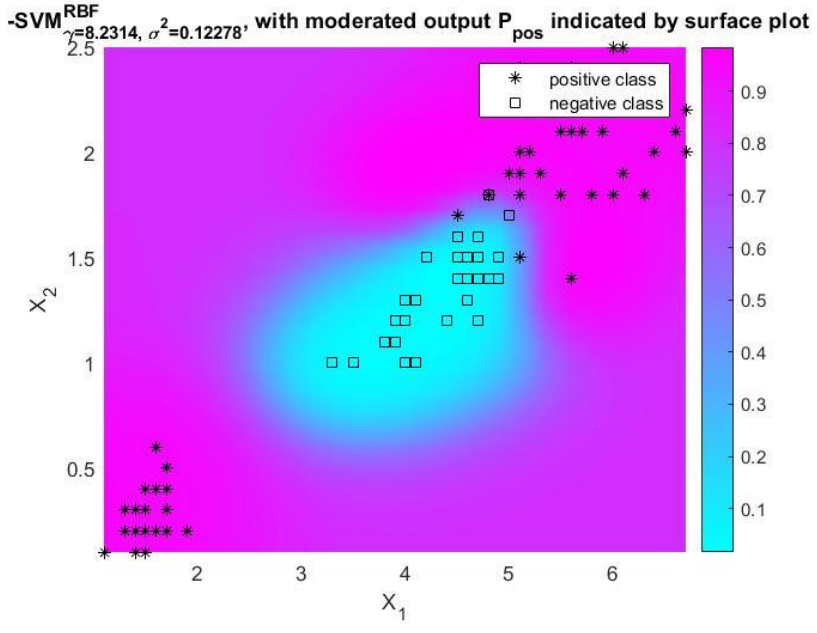


Figure 48 γ=8.2314, σ²=0.12278

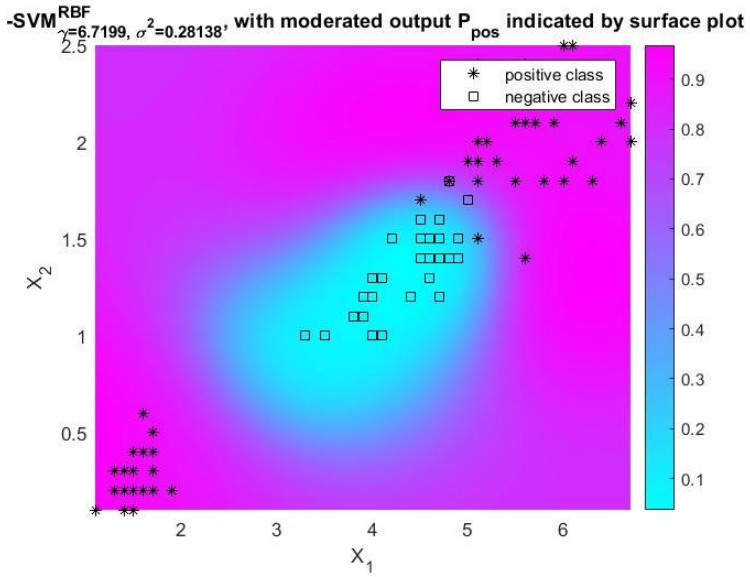


Figure 49 - γ=6.7199, σ²= 0.28138

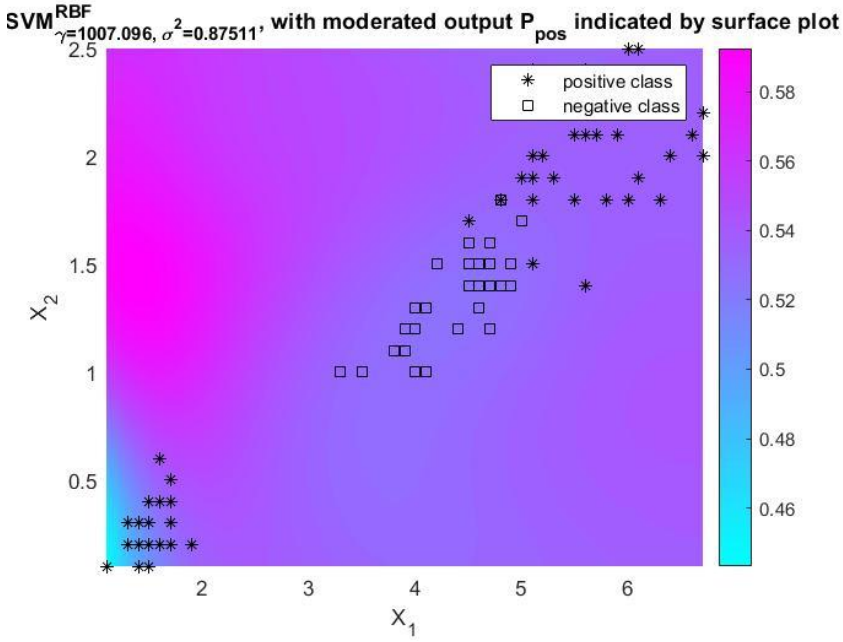


Figure 50 - γ=1007.096, σ²=0.87511363

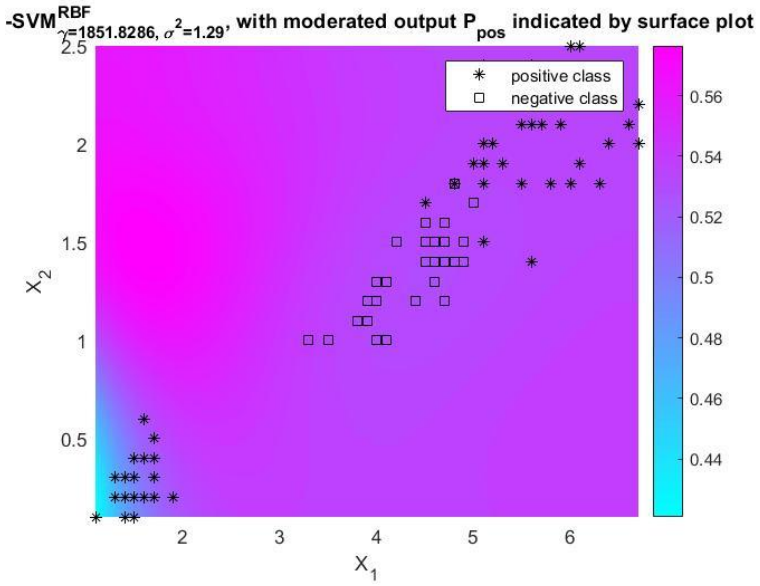


Figure 51 - γ= 1851.8286, σ²= 1.2899917

1.4 Homework problems

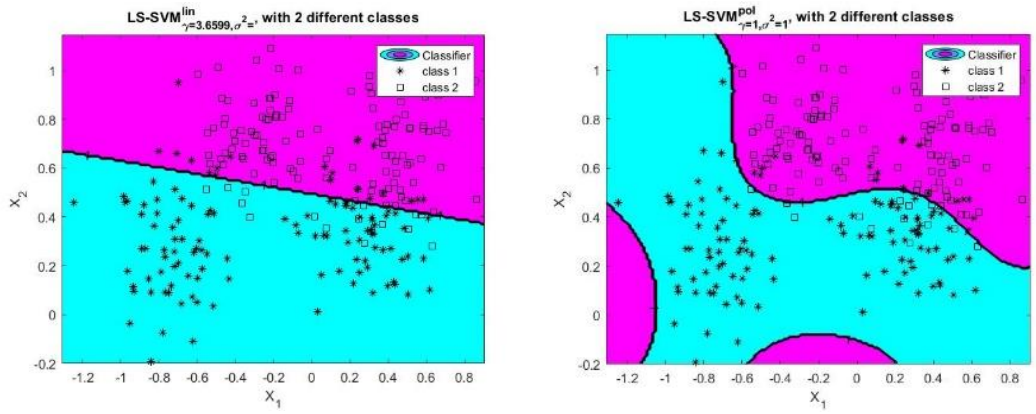
1.4.1 The Ripley Dataset

Figure 52 - Illustration of the Ripley training dataset

The Ripley Dataset is considered in this part. As it is 2-dimensional, it can be easily visualized in Fig. 52 (the training dataset). One may observe in fig. 52 that the Ripley data set is a non-linearly separable data set due to overlapping distributions. First, LS-SVM is introduced by means of classification as a linear model. The Ripley Data set consists of two classes where

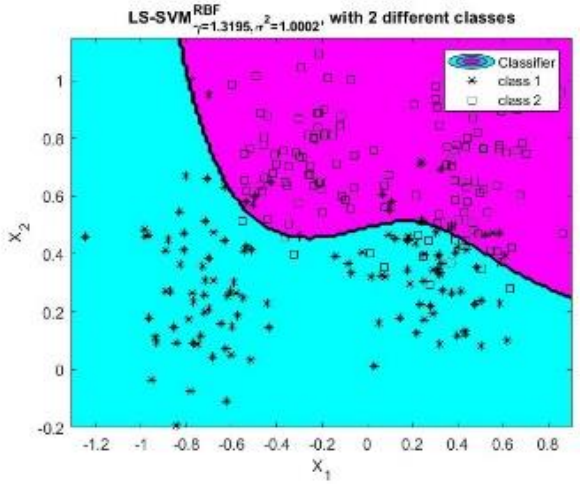
the data for each class is generated by a mixture of two normal distributions. Thus, a linear kernel may not be a suitable choice for the classification task of the data set, but this is investigated further on. Also, an LS-SVM model is built with a polynomial kernel of degree=5 and t=1 and finally an LS-SVM model with RBF kernel. For all the cases suitable tuning parameters are estimated for the LS-SVM model.

Figure 53 illustrates the classification task in all these three cases. The error rate in the case of a linear kernel is evaluated and is equal to 10.80%, while in the case of a polynomial kernel is equal to 10.20%. Finally, the error rate in the case of an RBF kernel is equal to 9.30%, thus only very small error improvement occurs in this case, comparing it with the error rates of the previous models. So, it is observed that performance does not improve substantially in the case of RBF kernel, but it is a more suitable choice than the other kernels, since it improves the performance of the model. This is something that is also visible in Fig. 54 where the ROC curves of each case are presented. Additionally, it is worth mentioning that when different starting points are tested the performance differs negligibly.



(a) Linear Kernel

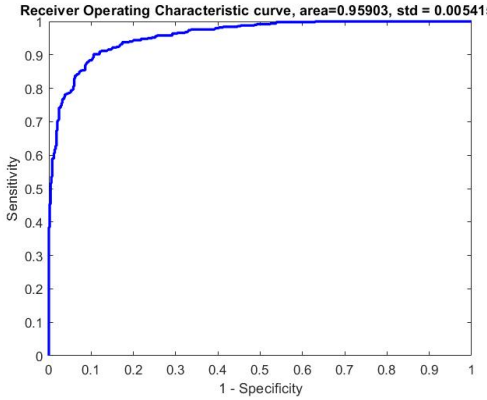
(b) Polynomial Kernel



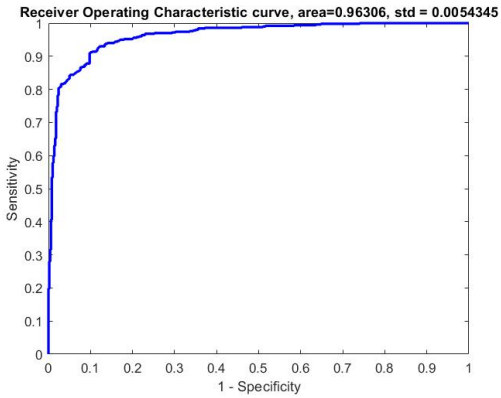
(c) RBF Kernel

Figure 53 - Illustration of the LS-SVM classifier on the Ripley binary classification data set: (a) Decision boundary for LS-SVM with a linear kernel (error rate is equal to 10.80%.), (b) Decision boundary for LS-SVM

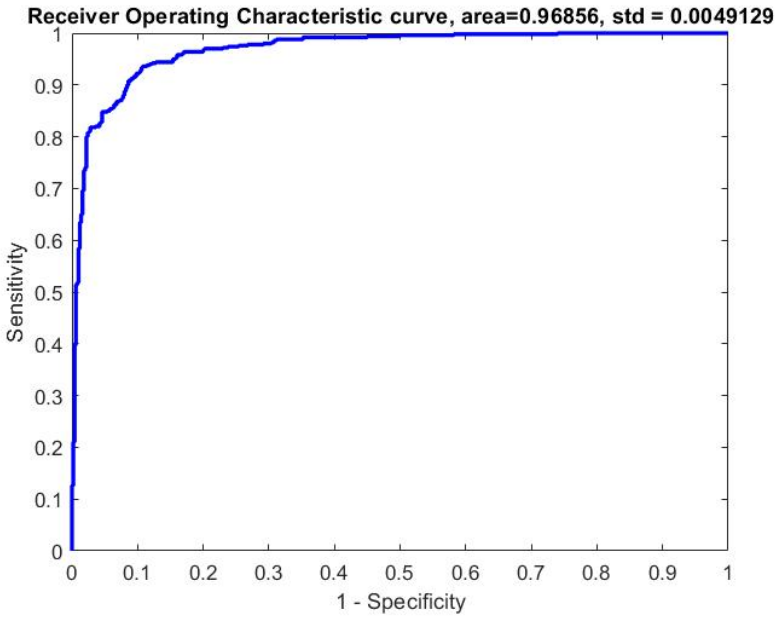
with a polynomial kernel of degree 5 (error rate is equal to 10.20%), (c) Decision boundary for LS-SVM with a well tuned RBF kernel (error rate is equal to 9.30%).



(a) Linear Kernel



(b) Polynomial Kernel

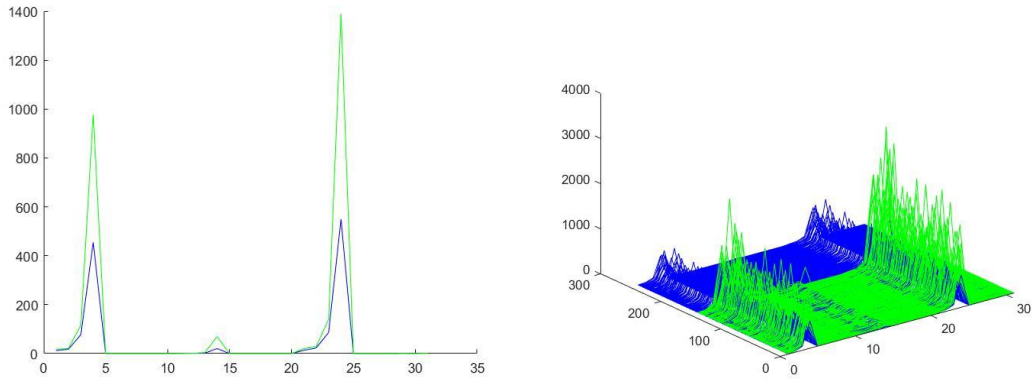


(c) RBF Kernel

Figure 54 - Illustration of the ROC curve for LS-SVM classifiers trained on Ripley dataset, (a) Linear kernel (b) Polynomial kernel of degree 5, (c) RBF kernel.

1.4.2 The Breast Cancer Dataset

The breast cancer dataset is shown in Figure 55. One may observe that the mean values corresponding to each class are distinct which is a good indication of the result the classification task will give.



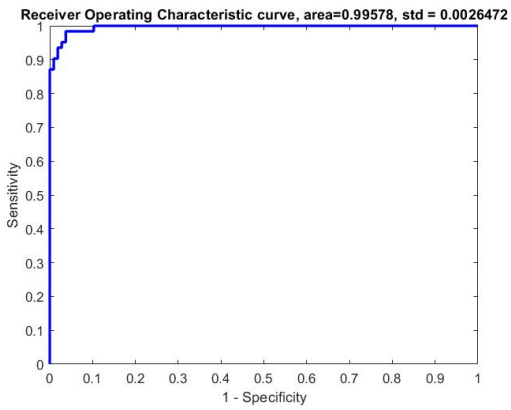
(a) - Mean Values plot

(b) - 3D-plot of the breast cancer dataset

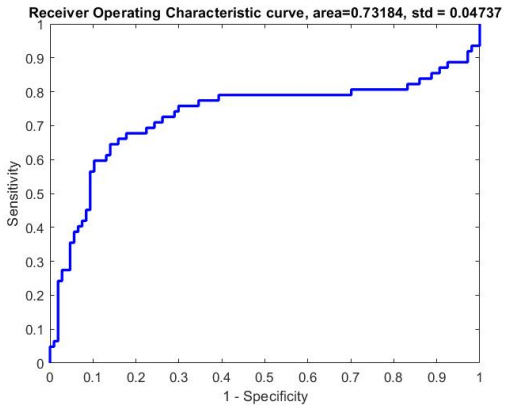
Figure 55 - Breast Cancer Dataset plot (a) Plot mean values of each class for training dataset (b)3D-plot of values of each class for training dataset

The Breast Cancer data set is analyzed within the context of classification. LS-SVM using linear kernel performs quite well, with error rate equal to 4.14%. The performance improves when using tuned RBF kernel with error rate equal to 2.37%. However, when using polynomial kernel (of degree 5) the performance is dramatically reduced with error rate equal to 23.08%. One may also observe that by using different starting points for the tunelssvm() routine the performance differs negligibly.

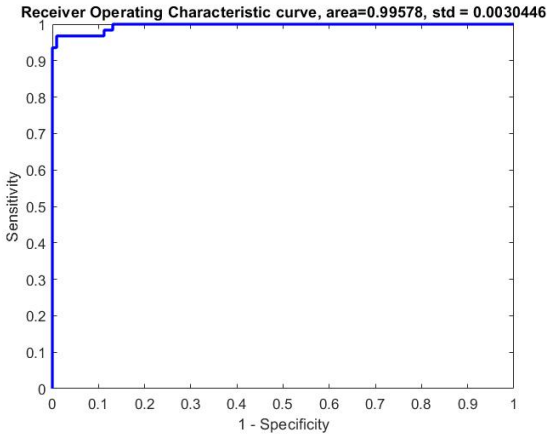
The receiver operating curve is illustrated in figure 56 for the LS-SVM classifier on the breast cancer database for linear, polynomial and RBF kernel. It may be observed in the receiver operating curve plots (figure 56) used to compare the models that there is no significant difference in performance between the model using linear and the one using the RBF kernel. Instead, the model using polynomial kernel has a really bad performance as it is also illustrated in the ROC curve (fig. 56 (b)).



(a) – ROC – linear kernel



(b) – ROC – polynomial kernel

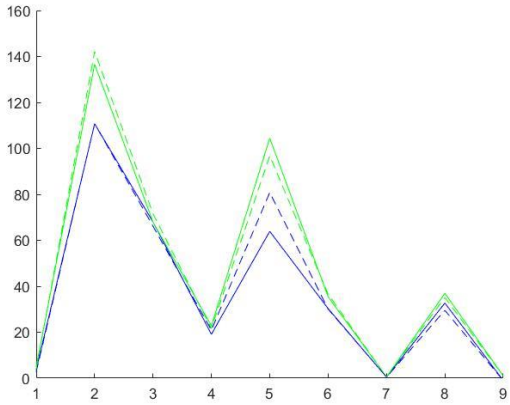


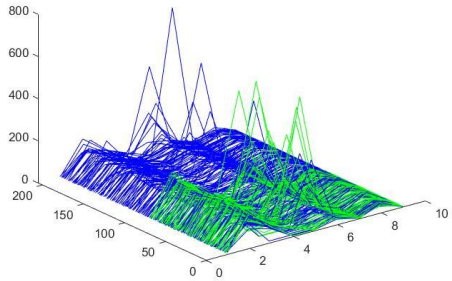
(c) – ROC – RBF kernel

Figure 56 - ROC curve for LS-SVM classifiers trained on Breast Cancer dataset

1.4.3 The Diabetes Dataset

The diabetes dataset is considered in this part (Figure 57). One of the first observations made here is that LS-SVM with linear kernel classifies the data with error rate equal to 21.43%, meaning that such a model is not sufficient for the classification. Furthermore, LS-SVM with RBF kernel gives similar results. In the latter case the error rate corresponding to the number of misclassifications is equal to 23.81%. Finally, LS-SVM with polynomial kernel has even worst performance than all the previous mentioned, with error rate equal to 38.69%.

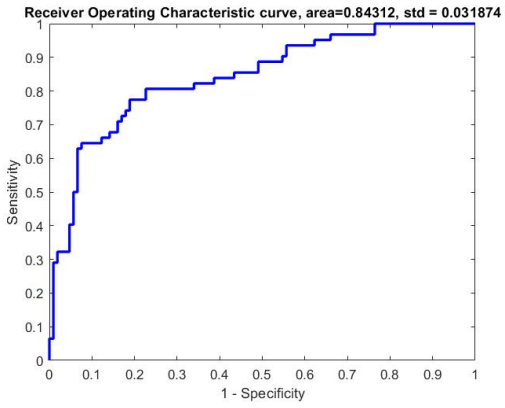


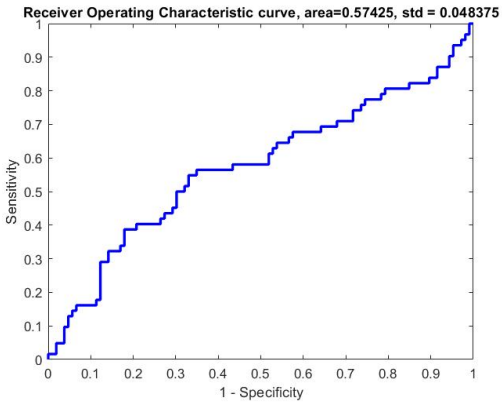


(a) - Mean Values plot **(b) - 3D-plot of the breast cancer dataset**

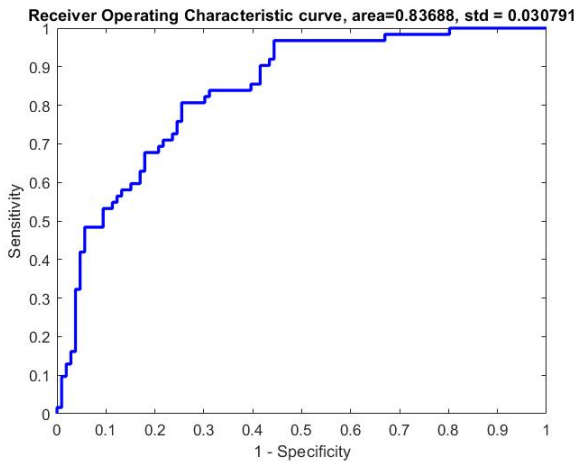
Figure 57 - Diabetes Dataset plot (a) Plot mean values of each class, for both training and test dataset. The test data are represented with '- -' while the training data with '-' (b)3D-plot of values of each class for training data

Subsequently the ROC curve is considered for the Diabetes dataset for the previously mentioned models (Fig. 58). Looking at the ROC Curves, one may observe that the performance is moderate in all cases, with that of the polynomial kernel dropping even more (Fig. 58-(b)).





(a) – ROC – linear kernel **(b) – ROC – polynomial kernel**



(c) – ROC – RBF kernel

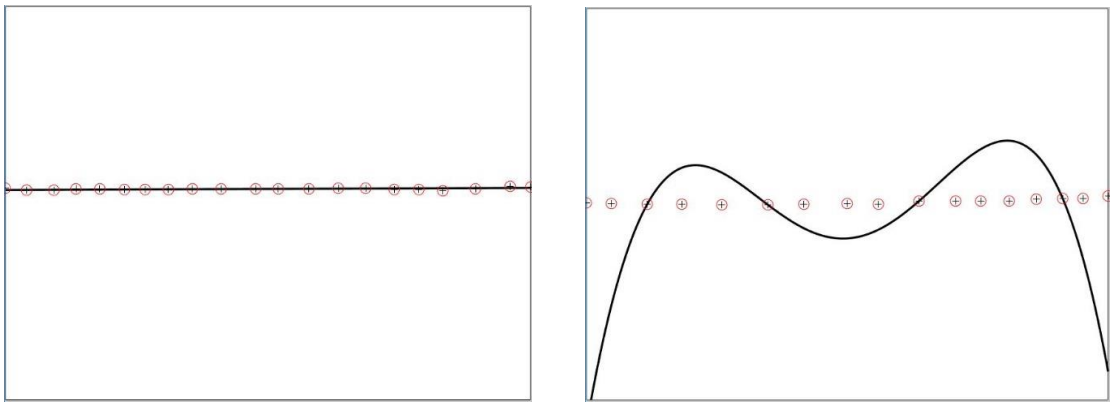
Figure 58 - ROC curve for LS-SVM classifiers trained on Diabetes dataset

If one considers the ROC curves as performances measure it can be observed that the performance of the linear kernel is a bit better comparing with that of the other two kernels, although, in general, the performance for all classifiers was not as good as with the previous datasets. This might be probably, due to overlaps between the two classes. Therefore, due to this moderate performance, one can suggest that this methodology is not perfectly suited for the Diabetes Dataset.

Exercise Session 2: Function Estimation and Time Series Prediction

2.1 Support Vector machine for function estimation

It may be observed that in the case of a linear dataset the most suitable option for regression is to use a linear kernel, whereas the use of quadratic kernels namely RBF and polynomial is prohibitive –figure 59.



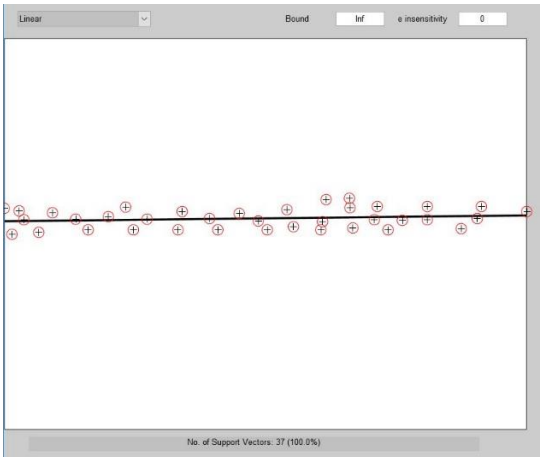
(a) Linear kernel (b) RBF kernel

Figure 59 - Illustration of SVM regression for a linear dataset (a) with linear kernel (b) with RBF kernel

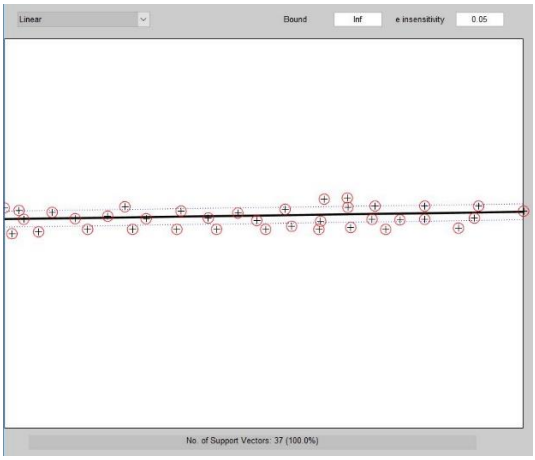
The value ϵ is the accuracy that one requires for the function approximation. When one chooses to increase the value of ϵ, then the boundary of the ϵ-tube will become larger and the number of support vectors is likely to drop, thus sparsity is introduced. The value of the parameter 'bound', the value of the tuning parameter c otherwise, controls how much we tolerate misclassifications, or in other words how much we tolerate that certain training data points are located outside of the so-called ε-tube. So, the value of c decides how much emphasis someone puts on the flatness of the function versus tolerating how many training data points are located outside of the ε-tube. This regularization term controls the flatness of the function, so by minimizing the regularization term, the result of the function estimation becomes flatter.

Figure 60 illustrates the influence of changing ϵ for a dataset of around 37 data points with respect to the number of support vectors using the SVM Matlab toolbox. An SVM function estimator with linear kernel for different ε values (0, 0.05, 0.1, 0.15, 0.2, 0.5) and the bound

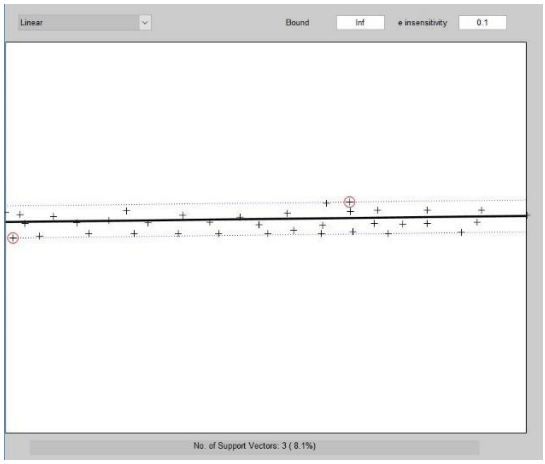
equal to Inf. is used. The insensitive ϵ cost function suggests that vectors within the tube should be viewed as classified correctly.



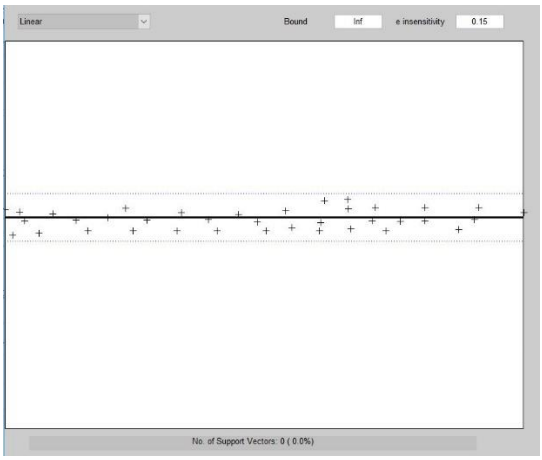
(a) SVM output for ε=0, Bound=Inf
(No of support vectors=37)



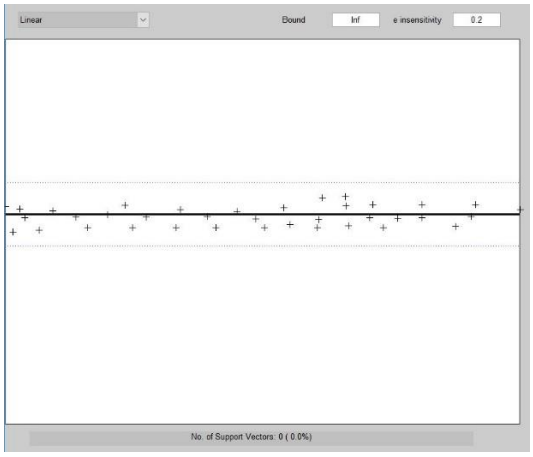
(b) SVM output for ε=0.05, Bound=Inf
(No of support vectors=37)



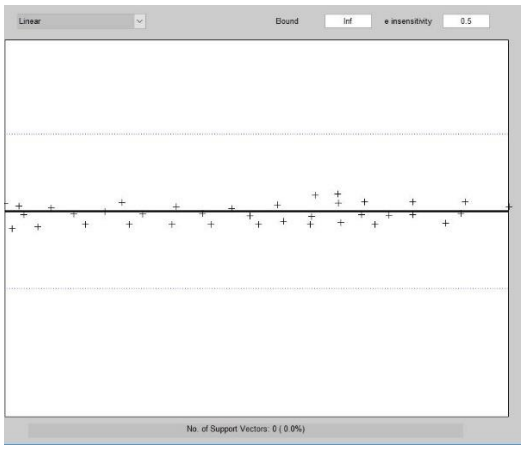
(c) SVM output for ε=0.1, Bound=Inf
(No of support vectors =3)



(d) SVM output for ε=0.15, Bound=Inf
(No of support vectors = 0)



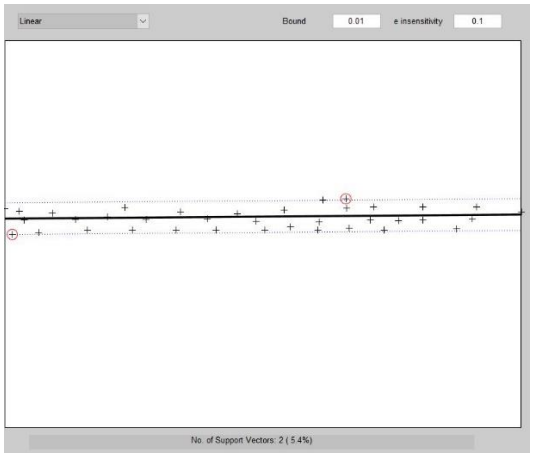
(e) SVM output for ε=0.2, Bound=Inf
(No of support vectors = 0)



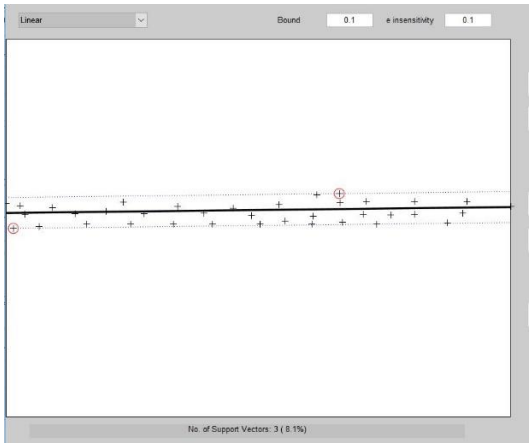
(f) SVM output for ε=0.5, Bound=Inf
(No of support vectors = 0)

Figure 60: Illustration of SVM regression with Linear kernel, for bound= Inf. and different values of Vapnik ϵ insensitive loss function: (a) ε=0, (b) ε=0.05, (c) ε=0.1, (d) ε=0.15, (e) ε=0.2, (f) ε=0.5 (e-tube boundary is described by the dashed line). It may be observed the bigger the ϵ the fewer the number of the selected support vectors. The insensitive ϵ cost function suggests that vectors within the tube are classified correctly therefore those vectors are not included in the solution.

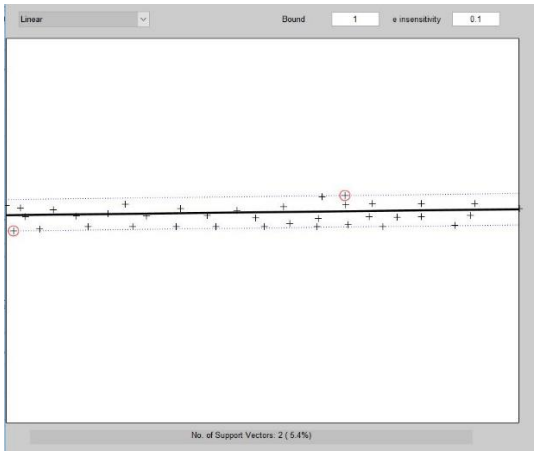
Figure 61 illustrates the influence of changing the regularization constant c (Bound) for the same dataset, with respect to the number of support vectors using the SVM Matlab toolbox. While experimenting, we notice some change in the angle formulated by the estimated function, as the bound affects the flatness of the function, but due to the nature the existing dataset it is not so visible. This is more visible, however, in another example with a different dataset, as it is illustrated in the Figure 62.



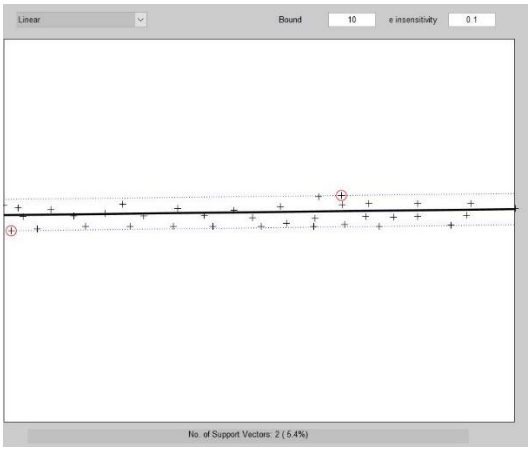
(a) SVM output for ε=0.1, Bound=0.01
(No of support vectors=2)



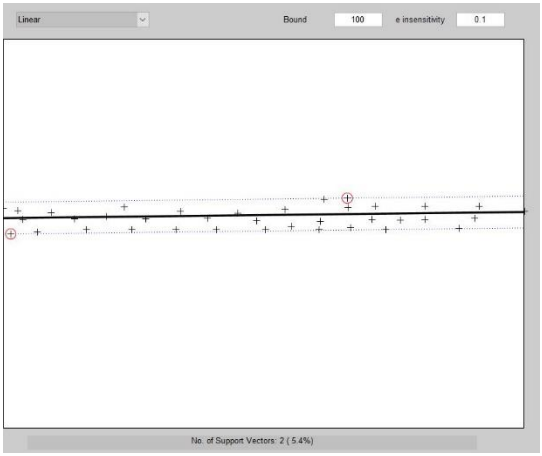
(b) SVM output for ε=0.1, Bound=0.1
(No of support vectors=3)



(c) SVM output for ε=0.1, Bound=1
(No of support vectors=2)



(d) SVM output for ε=0.1, Bound=10
(No of support vectors=2)

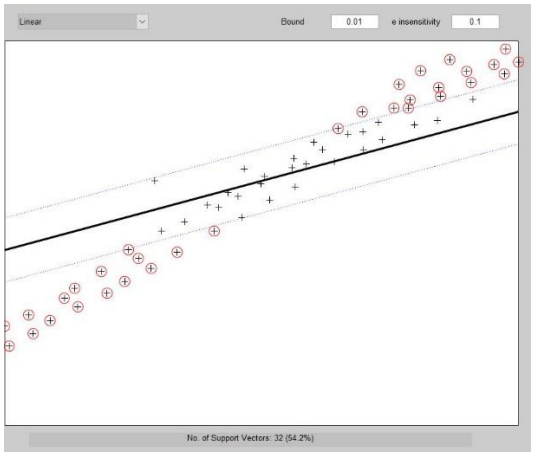


(e) SVM output for ε=0.1, Bound=100
(No of support vectors=2)

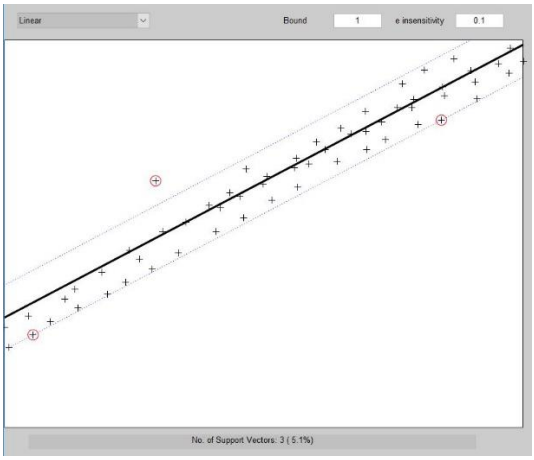
Figure 61: Illustration of SVM regression with Linear kernel, for ε=0.01 and different values of bound: (a) bound=0.01, (b) bound=0.1, (c) bound=1, (d) bound=10, (e) bound=100.

So, in Figure 62, the influence of the value of ε and Bound is a bit more clear. In more details, someone can see that in the following figures as the value of the ε increases the boundary of the ϵ-tube becomes larger, so that more data points to be located inside the ε-tube area. At the same time the number of support vectors drops. However, if ε grows too big then the

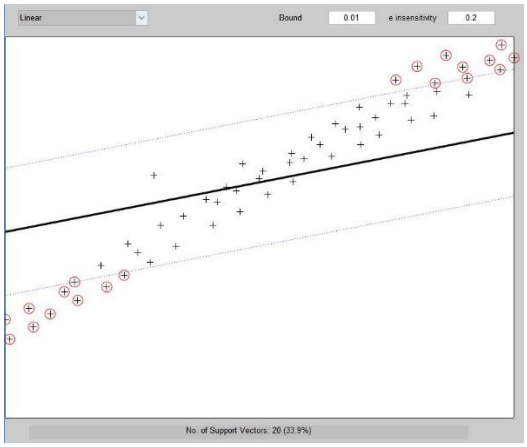
model is not able to capture the details of the underlying function. On the other hand, someone can see that when the value of the parameter bound is too small, it flattens the result of the function. The larger the bound, the more it tries to fit outliers (fig. 62(b)). The regularization parameter (bound) penalizes the vectors outside the ε-tube. It is important to note that in the case where the regularization parameter is set to very high values the model then tries to fit data perfectly without considering complexity and then there is the danger of overfitting. Having a good combination of these two parameters though, a good result can be achieved (Fig.62 (d)).



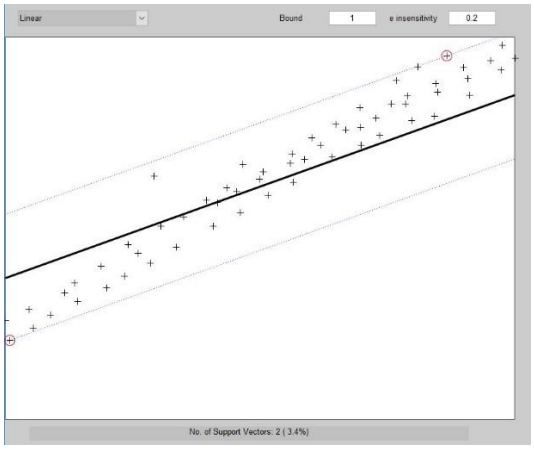
(a) SVM output for ε=0.1, Bound=0.01
(No of support vectors=32)



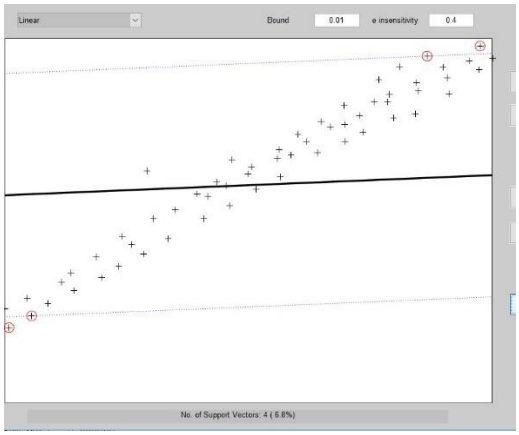
(b) SVM output for ε=0.1, Bound=1
(No of support vectors=3)



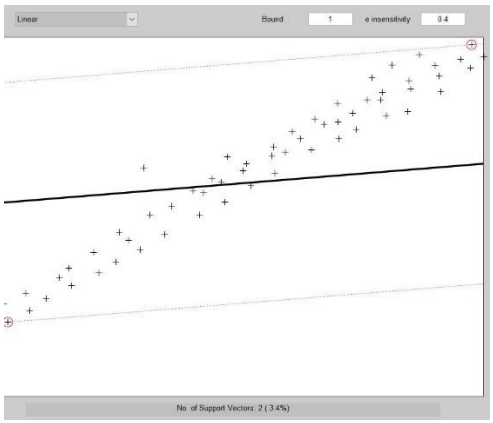
(c) SVM output for ε=0.2, Bound=0.01
(No of support vectors=20)



(d) SVM output for ε=0.2, Bound=1
(No of support vectors=2)



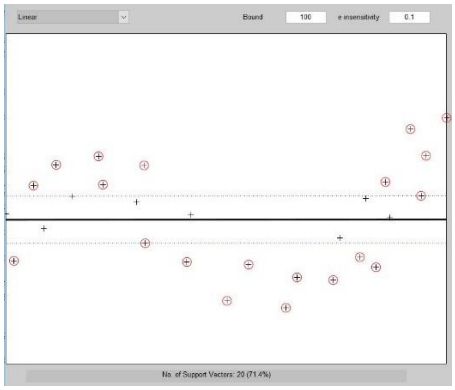
(e) SVM output for ε=0.4, Bound=0.01
(No of support vectors=4)



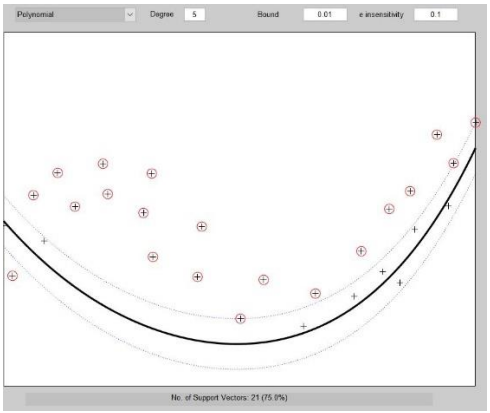
(f) SVM output for ε=0.4, Bound=1
(No of support vectors=2)

Figure 62: Illustration of SVM regression with Linear kernel, different values of ε and bound

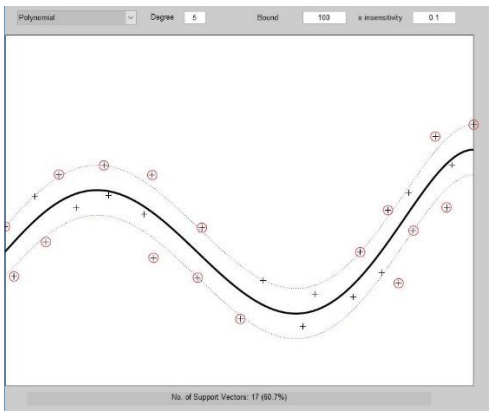
Next, a more challenging dataset is constructed and the results of the SVM function estimator with different kernels are presented later on. The value of bound is selected equal to 100 (apart from the Figure 63 (b) where bound=0.01) and the value of the parameter ε is equal to 0.1. As someone can notice from the following figures, the SVM with the linear kernel is not able to fit the more complex non-linear underlying function (Figure 63 (a)). In contrast, the SVM polynomial kernel of degree equal to 5 produces a very good estimation of the function, but only for the case where the proper regularization constant is used (Fig. 63 (c)). Without the proper regularization, it tends to interpolate through the noisy data points (overfitting), so the selection of the hyperparameter c is very important and leads to a good function estimation in this case. Finally, concerning the results of the SVMs with RBF kernel, we should highlight the importance of choosing the right value of sigma. When sigma is large (Fig. 63 (f)), the decision boundary tends to become linear, while when it is too small (Fig. 63 (d)) the decision boundary becomes highly nonlinear. For the proper value of sigma, though (Fig. 63 (e)), the model is able to generate a good result and fit the function.



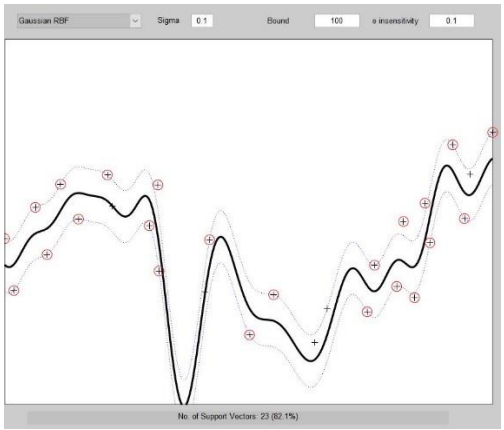
(a) Linear Kernel



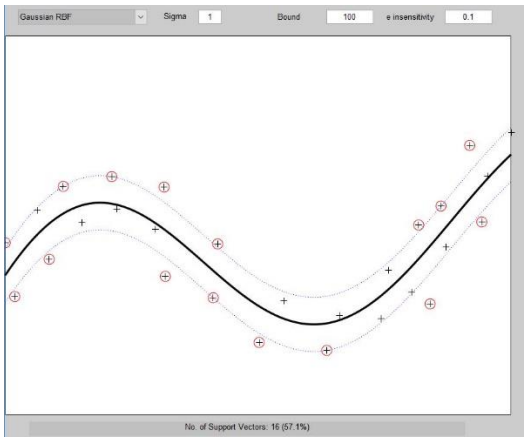
(b) Polynomial Kernel, degree=5, bound=0.01



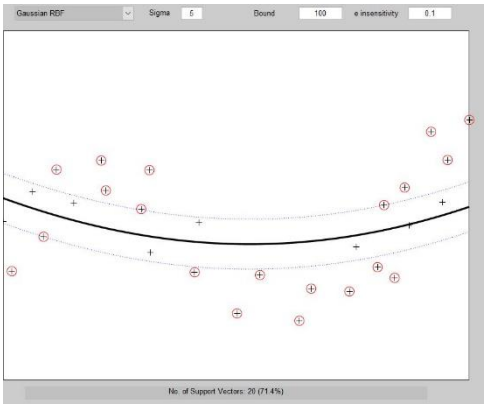
(c) Polynomial Kernel, degree=5, bound=100



(d) RBF kernel – sigma= 0.1



(e) RBF kernel = sigma = 1



(f) RBF kernel - sigma=5

Figure 63 - SVM with different kernels for function estimation. bound = 100, ε = 0.1.

In comparison with the classical least square fit, SVM regression formulation contains more tuning parameters. In the case of an RBF kernel sigma (σ), bound (c) and ϵ are to be considered as additional tuning parameters which do not follow as the solution to a QP problem but also need to be determined in some way. Least square fit solves a set of linear equations whereas SVM regression is a convex optimization problem.

2.2 A simple example: the sinc function

2.2.1 Regression of the sinc function

The LS-SVM classifier is constructed here by means of an RBF kernel. In this part the following distribution is considered. The training data have been generated by a sinc function.

Subsequently, the results on the test set are discussed when selecting different values of gamma and sigma. On may observe in figure 64 the effect of selecting different values of σ^2 (e.g., 0.01, 1, 100) and gamma (e.g., 10, 10^3, 10^6). Considering the case of σ^2, the results are reasonable considering that in the case of RBF kernel for large values of sigma the approximated function tends to become linear and for small values of sigma the approximation function tends to become highly nonlinear. One may suggest here that small values of sigma give a better fit.

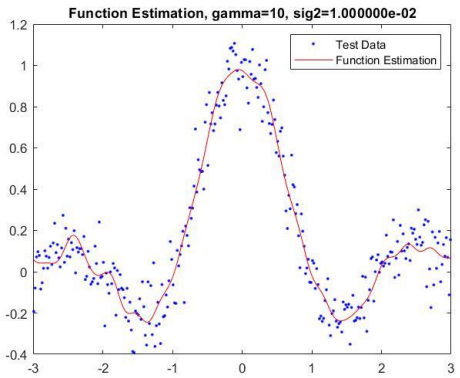
Least Squares Support Vector Machines for function estimation formulate the following optimization problem:

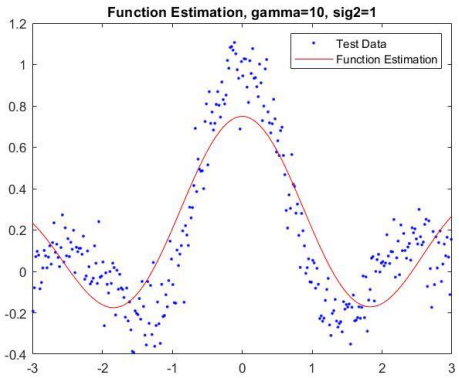
$$\min_{w,b,e} \mathcal{J}(w,e) = \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{k=1}^{N} e_k^2$$

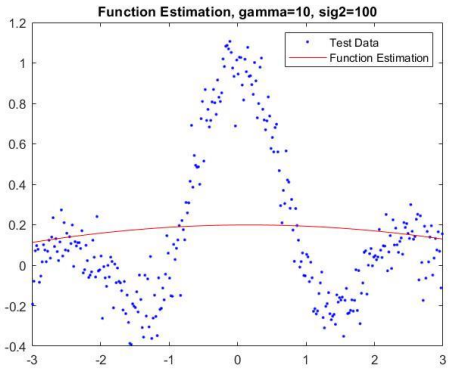
This is subject to equality constrains:

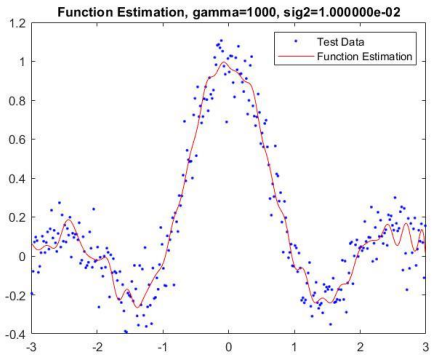
$$y_k = w^T \varphi(x_k) + b + e_k, \quad k = 1, ..., N$$

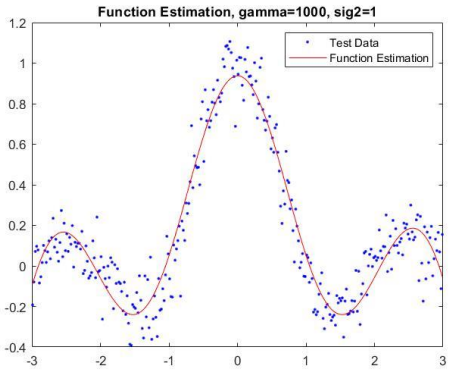
The hyper-parameters of the LS-SVM model are the parameter sigma of the Gaussian kernel and the gamma (γ) regularization factor. The problem of function approximation consists in the determination of the relationship between set of x inputs and one single output y. Any estimation of the relationship between input-output pairs goes through a compromise between a low learning error and a smooth model. In the case of LS-SVM this compromise is implemented through the choice of an adequate value of γ. If the value of γ is set too large the model will overfit the data. Still the value of γ should be set as large as possible; a too small value of γ would simple mean that the model does not fit the learning data. Figure 64 illustrates the effect of selecting a range of different values of γ (γ=10, 10^3, 10^6).

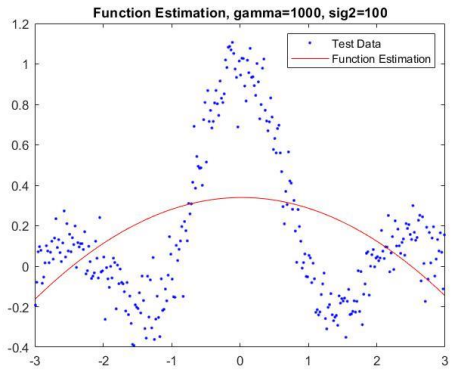


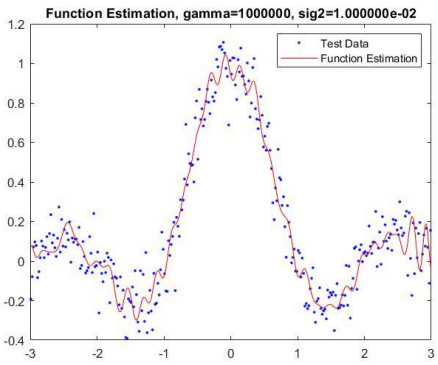


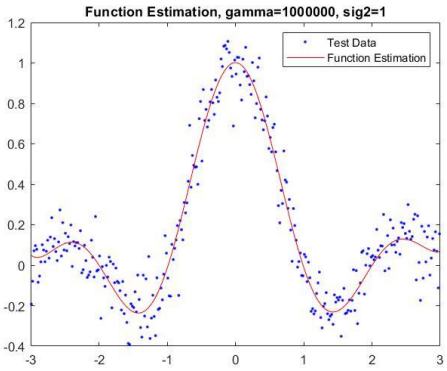

Function Estimation, gamma=10, sig2=100
Test Data
Function Estimation


Function Estimation, gamma=1000, sig2=1.000000e-02
Test Data
Function Estimation


Function Estimation, gamma=1000, sig2=1
Test Data
Function Estimation


Function Estimation, gamma=1000, sig2=100
Test Data
Function Estimation


Function Estimation, gamma=1000000, sig2=1.000000e-02
Test Data
Function Estimation


Function Estimation, gamma=1000000, sig2=1
Test Data
Function Estimation

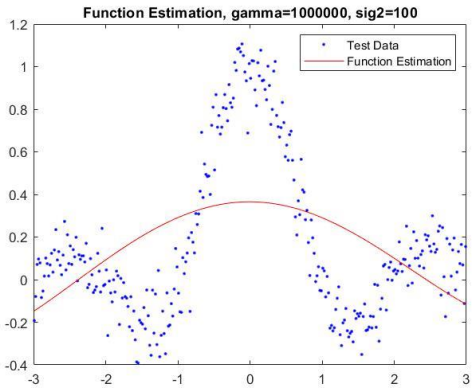


Figure 64: LS-SVM approximation of a sinc function with RBF kernel. The results of function estimation are plot for different values of sigma2 (e.g., 0.01, 1, 100) and gamma (e.g., 10, 10^3, 10^6).

In Table 14, it is stated also the mean squared error for every combination of γ and σ^2 .

mse	σ^2=0.01	σ^2=1	σ^2=100
γ=10	0.009965	0.035383	0.134792
γ=10^3	0.010891	0.011874	0.117083
γ=10^6	0.011796	0.009575	0.111207

Table 14 – Mean Squared Error for every combination of γ and σ^2

It is important to note here that not only one pair of optimal hyper-parameters exist but several ones corresponding to different local minima.

Next, we proceed with the process of optimizing the γ and σ^2 hyper-parameters with the Matlab tunelssvm procedure. One may observe by running the tunelssvm command several times that there exist several sets of optimal hype-parameters values (table 15, table16). Gridsearch is a two dimensional optimization method based on exhaustive search on a limited range .It is the simplest algorithm to determine the minimum of a cost function with possibly multiple optima by taking a grid over the parameter space in order to pick the minimum. This procedure iteratively zooms to the candidate optimum. Simplex is an alternative optimization function which also finds local minima. Both of them find good results but probably not the optimal solution. Furthermore, one may observe that grid search approximation function is slower.

Simplex Algorithm:

γ	σ^2	mse
13.801	0.333123	0.010521
15.9609	0.306587	0.012236
46.787	0.404244	0.009210
337.0038	0.04667186	0.010884

Table 15 - Set of optimal hyper-parameters tuned with the method of cross validation using the simplex algorithm and the corresponding mean squared error

Grid Search Approach:

γ	σ^2	mse
23.9043	0.374921	0.010910
2449.9853	0.62165778	0.011638
54.3955	0.344241	0.010484
24.204	0.251233	0.011129

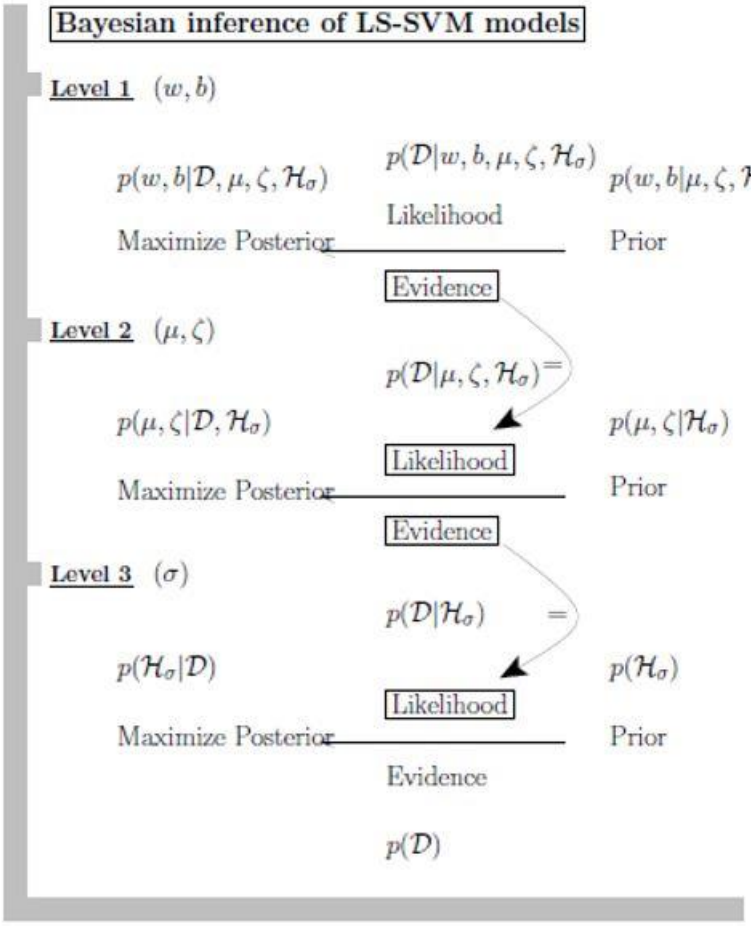
Table 16 - Set of optimal hyper-parameters tuned with the method of cross validation using a grid search approach and the corresponding mean squared error

2.2.2 Application of the Bayesian framework

There are several ways, in general, to proceed in order to determine the parameters (γ,σ). A simple way is to work with a training set, validation set and test set, explore a meaningful grid of possible (γ,σ) combinations and select the values in such a way that they give the best perfomance on this validation set. But the results, in this case, might be too sensitive with respect to the chosen validation set. In a statistical sense, it is therefore better to do k-fold

cross-validation. An alternative approach to that, though, is the determination of the hyperparameters at higher levels by Bayesian inference.

Sunsequently, a Bayesian framework is used now to tune and to analyze the LS-SVM regressor. The purpose of this model is to optimize the posterior probabilities of model parameters with respect to the different levels of Bayesian inference. In the first level one optimize the support values a's and b's, in the second level one optimizes the regularization parameter γ and in the third level one optimizes the kernel parameter. In the case of the common RBF kernel the parameter is the bandwidth σ^2.The schematic visualization of the three levels principle describing the Bayesian framework is shown in schematic 1.



Schematic 1: Bayesian framework

By using the Bayesian framework, we can have uncertainty in the regression task, which is an expected value for the prediction. In next Figure someone can observe the function

estimation result by using the Bayesian framework and also the confidence interval. This one provides information about the uncertainty that we have for the prediction.

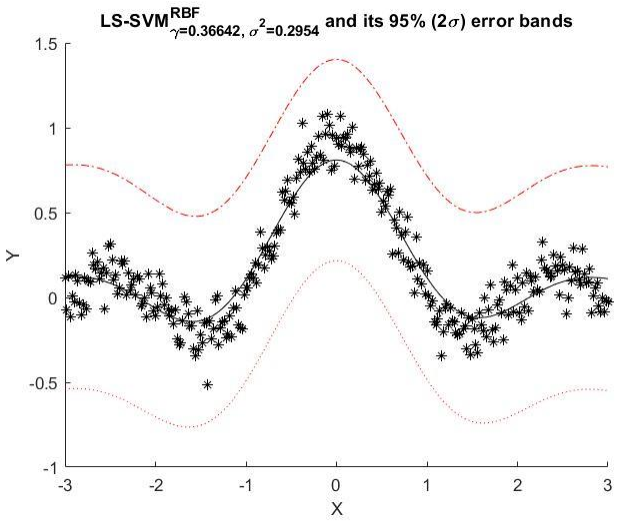


Figure 65 Illustrates the results of Bayesian framework applied to LS-SVM for function estimation

2.3 Automatic Relevance Determination

Additionally to parameter tuning, the Bayesian framework can also be used to select the most relevant inputs by Automatic relevance Determination (ARD). In this task, we are given a training set of fixed length feature vector (3 dimensions) along with target values. We use Bayesian framework criterion for backward selection for the three dimensional input selection task, and tuned γ and σ^2 parameter values. The results are presented in Figure 66. As someone can observe, the variable that most affects the minimization cost is X1. Therefore, X1 proves to be the most relevant input. This is also obvious after executing the Matlab bay_lssvmARD command, where the features are ranked in order ([1;3;2]). Also, someone can also notice that X2 and X3 is that combination of inputs that is most irrelevant, having the maximum error between all the combinations. The problem of feature space selection can be defined as finding relevant features among the original feature vector, with the purpose of increasing the accuracy of the resulting model or reducing the computational load associated with high dimensional problems. Indeed , in this example, we see that this is achived by keeping the X1 feature vector for example. In general, the approach using ARD for feature ranking can achieve a more compact feature set along with better performance that may prove to be useful in the case of a high dimensional feature set.

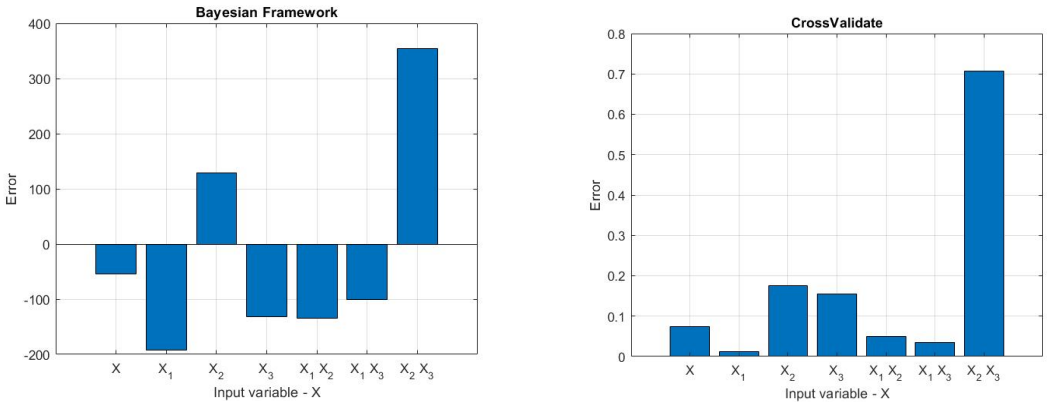
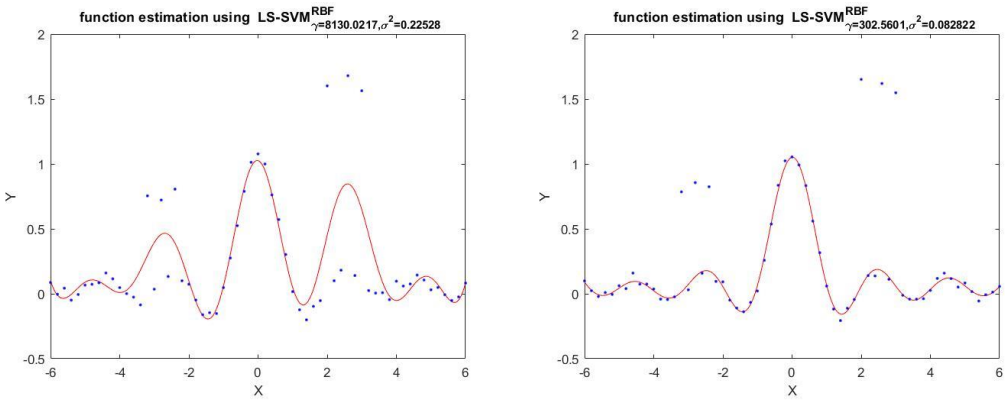


Figure 66: Automatic Relevance Determination (ARD) results.

Additionally, we also perform input selection in a similar way, by using the Matlab crossvalidate command instead of the Bayesian framework. Again, we check all the possible input combinations and perform the cross validation process. The results are presented in Figure 66. As someone can notice in this case, too, the selection of the most relevant input is again the X1 and the less relevant input combination is X2 and X3, exactly as it is calculated by the Bayesian framework. This means that the two frameworks are consistent.

2.4 Robust Regression

The case where the data is corrupted with non-Gaussian noise or outliers is now considered. In the presence of outliers that do not come from the same data-generating process as the rest of the data, least squares estimation is insufficient and can be biased. One may visualize the influence of the outlying data points in the case of regression with LS-SVM model, in the Figure 67(a) and notice that these outlying points make the model to avoid fitting the underlying function. Tuned parameters γ and σ^2 are used in the cases presented below.



(a) Non-Robust Version (b) Robust Version

Figure 67: Automatic Relevance Determination (ARD) results.

Subsequently, robust LS-SVM model is considered in order to accommodate outliers. Robust regression can be used in any situation in which someone would use least squares regression. It might be a good strategy since it is a compromise between excluding the outlying data points entirely (we have no compelling reason to exclude them from the analysis) from the analysis and including all the data points and treating all of them equally in regression. The regression estimation technique lessens the influence of outliers by reducing their weights. The output of robust LS-SVM regression model is shown in Figure 67(b).

In addition, attention is given to the models' predictive ability using the test of mean squared error (mse). Simulations indicate that mean absolute error (mae) can provide more accurate performance measure than mean square error (mse). Thus, in the case of robust regression, it should be preferred to allow model parameters to be estimated using the mae loss function. One may refer to the mean absolute error function as a more resistant loss function to outliers. On the contrary, the mean squared error may be more sensitive to the presence of outliers and thus may result in a biased optimization process -Figure 68.

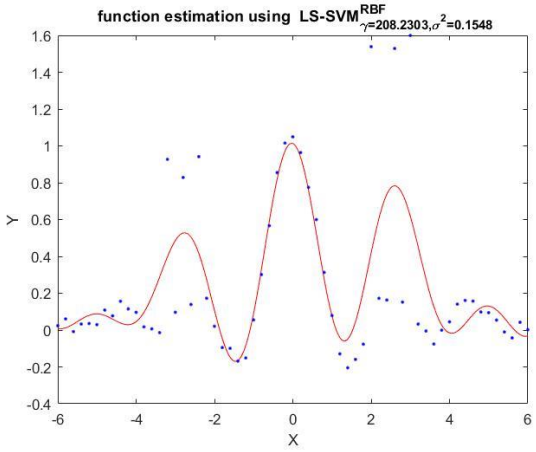
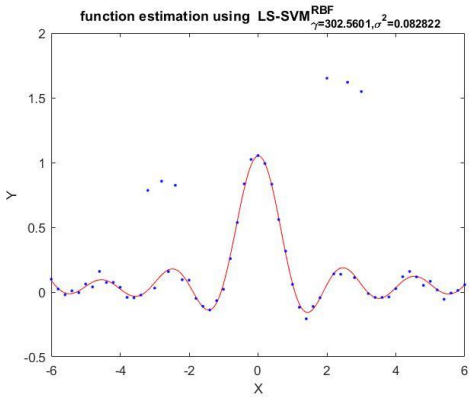
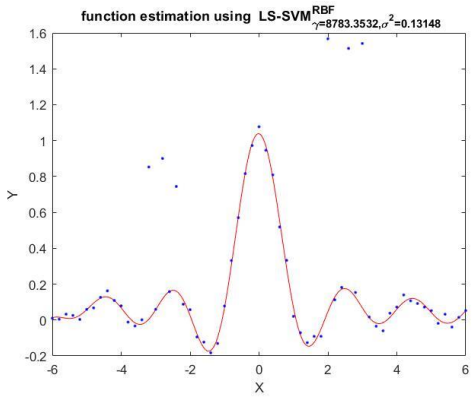


Figure 68: Output of robust regression model in the case of mean squared error

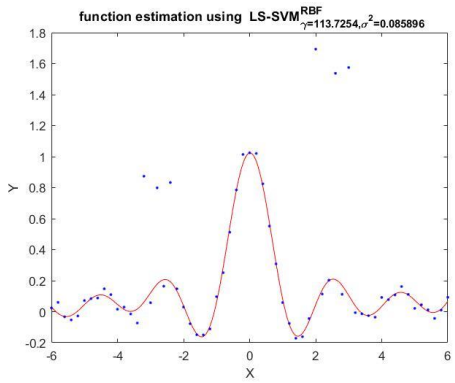
There are several alternatives weighting functions that can be used for robust regression. First, the behavior of huber weighting function is tested and then compared with some alternative ones (e.g. 'hampel', 'logistic', 'myriad') - figure 69. When comparing the results of all four weighting functions mentioned above, one may say that the parameter estimates coming from these different weighting methods differ; however all four of them perform well and there are not significant differences detected. There is though evidence that different functions have advantages and drawbacks. One example may be given where hampel function outperforms by mean of convergence time.



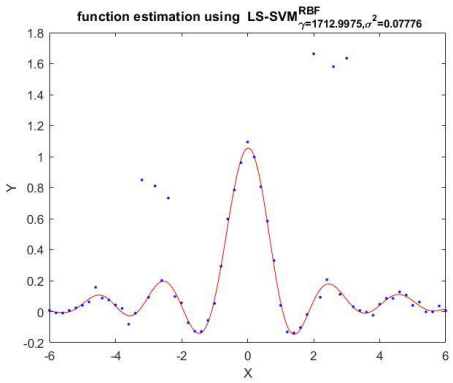
(a) Huber



(b) Hampel



(c) Logistic



(d) Myriad

Figure 69: Illustrates the output of robust regression model in the case of mean absolute error using different weighting functions

2.5 Homework problems

2.5.1 The Logmap Dataset

Time series prediction is studied in this part. Time series prediction is the use of a model to predict future values based on previously observed values. In this section, we apply time series prediction on the Logmap dataset. Initially, the parameters that are used are σ^2= 10, γ=10, order=10 and the results of applying time series prediction on Logmap dataset are presented in Figure 70(a). As someone can observe in this figure, the model is not able to predict properly the future values. For this reason, we optimize subsequently the parameters γ and σ^2 using crossvalidation and then using these optimized parameters we search for the optimal value of order by comparing the MSE on the test data for a specific choice of order each time. The MSE that we get for every option of order value, is presented in table 17. There, someone can see that the optimal value for the order is 23, as it gives the smallest mean squared error. Finally, we retrain the model with the tuned values and get the result presented in Figure 70(b). As we can observe, the results have been improved and now the model is closer to the reference.

order	mse
10	0.112838
11	0.246520
12	0.106346
13	0.110498
14	0.133478
15	0.074737
16	0.099732
17	0.084305
18	0.107396
19	0.116943
20	0.071063
21	0.075522
22	0.060878
23	**0.029880**
24	0.041127
25	0.055204
26	0.059514
27	0.051959

Table 17 – Illustrates the MSE on test data for various options of order value

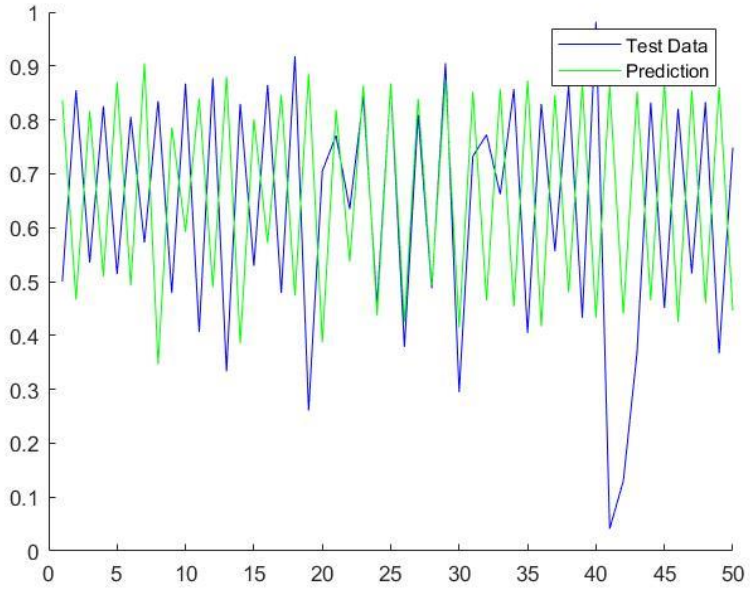


Figure 70 (a): Times series prediction on Logmap dataset, γ=10, σ^2=10, order=10

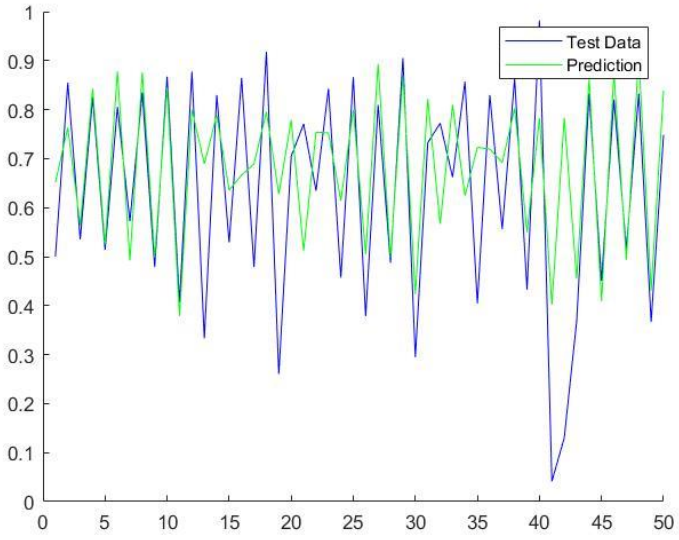


Figure 70 (b): Times series prediction on Logmap dataset for tuned γ, σ^2 and order.

2.5.2 The Santa Fe dataset

In this section, we apply time series prediction on the Santa Fe dataset. In figure 71 someone can see the visualization of this dataset, for both the training and the test data. Judging from these plots, the selection of order=50 is not a good choice for the AR model, as the abrupt changes in the time series take place after a long period of time.

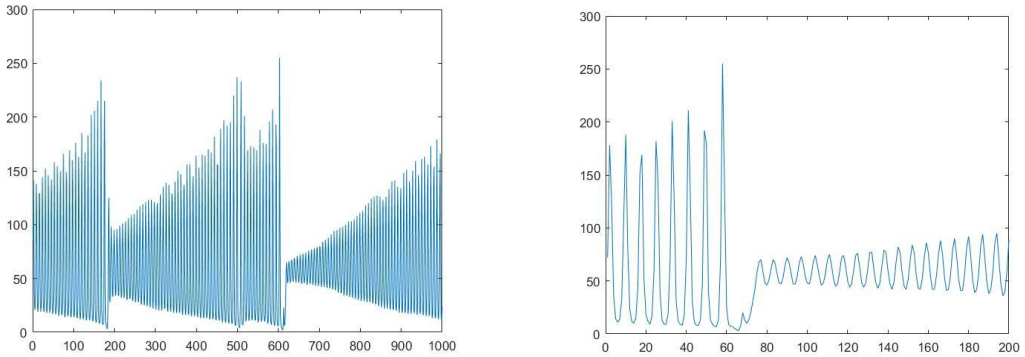


Figure 71: Visualization of the training (left figure) and test dataset (right figure) for the Santa Fe dataset.

Indeed, using order=50 in the utilized AR model, the performance of the prediction proves out to be bad (Figure 72).

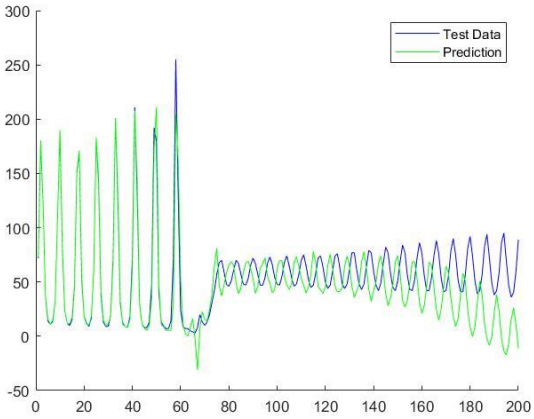


Figure 72: Times series prediction on Santa Fe dataset for order=50.

Then, we optimize the parameters and by comparing the MSE on the test data for different order values, we end up that order=100 could prove to be a good option, but not perfect (Figure 73).

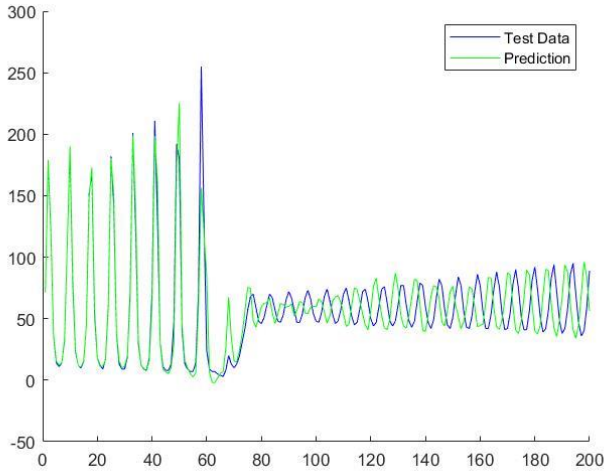
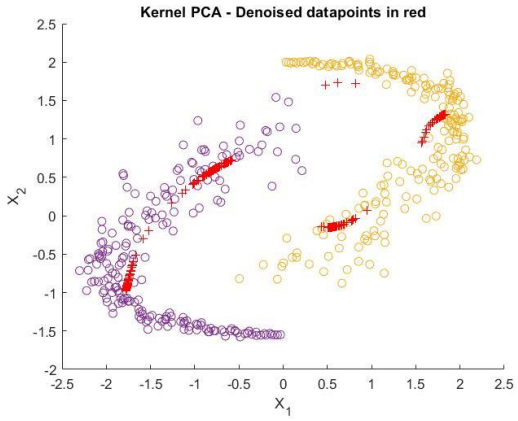


Figure 73: Times series prediction on Santa Fe dataset for order=100.

Exercise session 3: Unsupervised Learning and Large Scale Problems

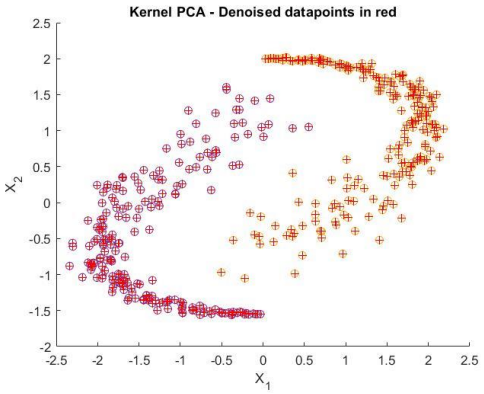
3.1 Kernel principal component analysis

In this part, a denoising example is introduced and further discussed by means of kernel PCA and linear PCA. Kernel principal components analysis projects observed data non-linearly into a high-dimensional feature space and then performs linear PCA in the feature space using the kernel technique. It may be considered as an extension of classical principal components analysis (PCA). Principal component analysis is a procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such way that the first principal component has the largest possible variance, and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal. The main difference when one applies kernel PCA is that the number of principal components may go beyond the dimensionality of the input space. In other words, the maximum amount of principal component that can be obtained using linear PCA corresponds to the dimensionality of the original input space. For the kernel PCA, the dimensionality can be higher, and is in fact completely independent of the dimensionality of the input space.



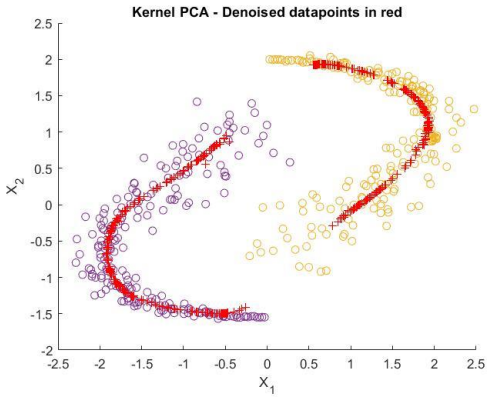
No. of Principal Components=2,

Lanczos approximation



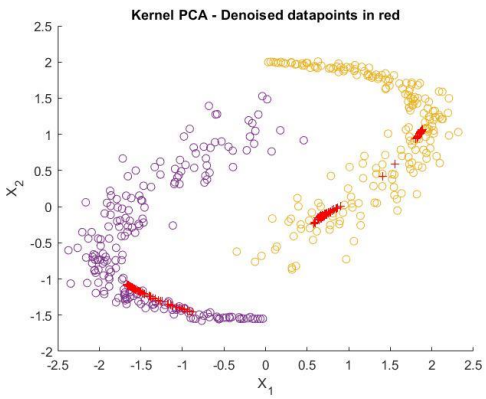
No. of Principal Components=2,

Nystrom approximation



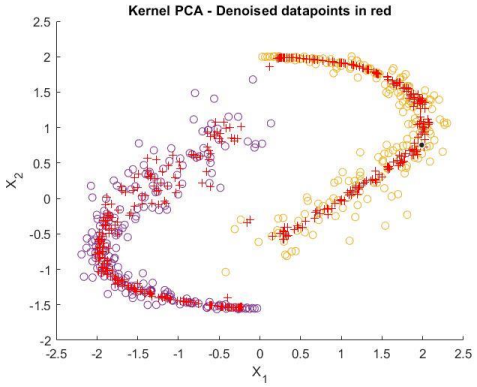
No. of Principal Components=6,

Lanczos approximation



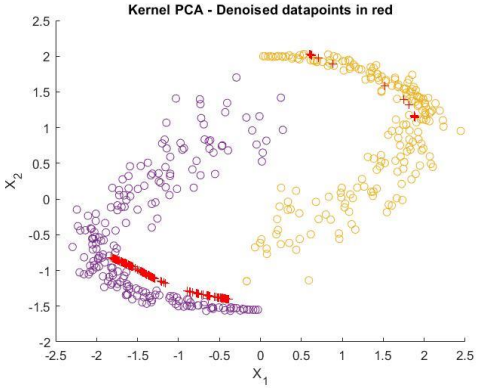
No. of Principal Components=6,

Nystrom approximation



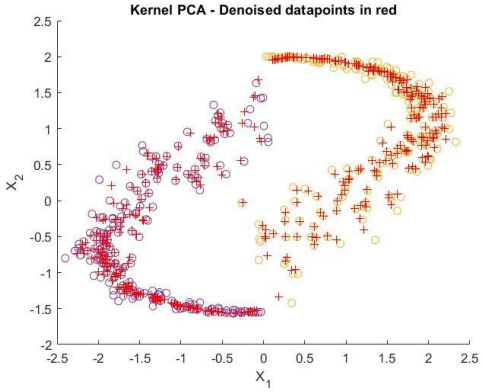
No. of Principal Components=12,

Lanczos approximation



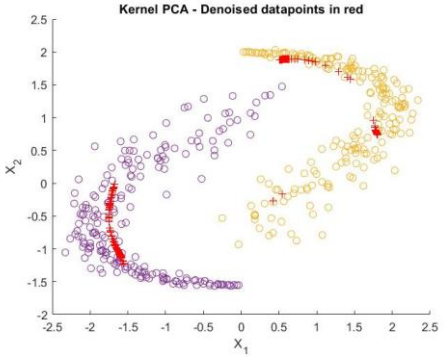
No. of Principal Components=12,

Nystrom approximation



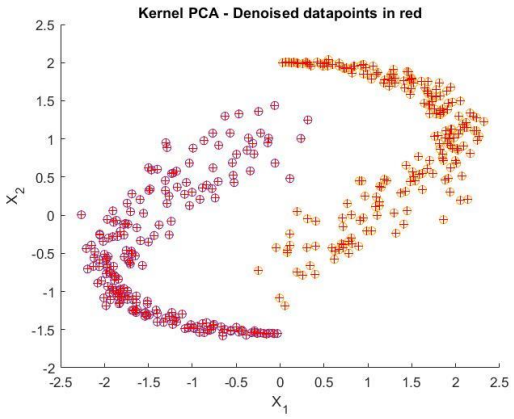
No. of Principal Components=18,

Lanczos approximation

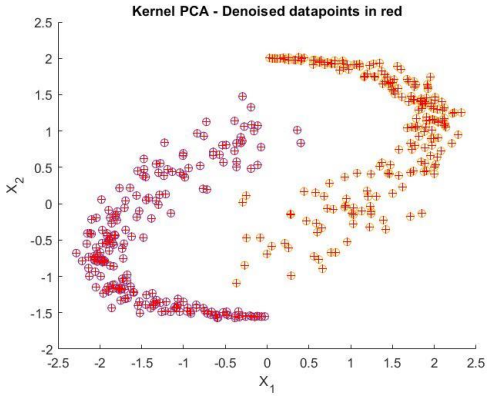


No. of Principal Components=18,

Nystrom approximation



No. of Principal Components=100,
Lanczos approximation



No. of Principal Components=100,
Nystrom approximation

Figure 74: Illustrates the original data set in ('o') and the denoised data in ('+') first in the case where Lanczos approximation is used and then in the case where Nystrom approximation is selected, for different number of principal components (2,6,12 and 18 respectively).

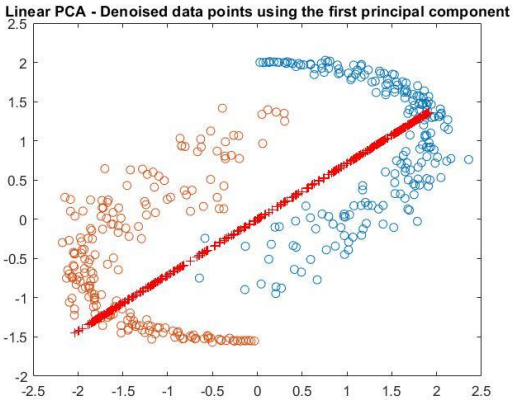


Figure 75: Illustrates the original data set in ('o') and the denoised data in ('+') by means of classical PCA analysis.

When denoising a data-set, the main goal is to preserve the underlying information without capturing the noise i.e a trade-off between noise suppression and preserving actual relations. After reconstruction, the data-set should appear without noise. This is done by reducing the dimension of the input dataset and by this mean keeping the important information and not the noise. One may observe in figure 75 that linear PCA is not sufficient for the given data set (does not represent the variance of the original set) whereas KPCA based denoising technique is a more efficient approach and outperforms linear PCA. However, when the number of principal components increase beyond the default value

(nb=6), the performance of the denoising method by means of KPCA drops and the reconstructed data set becomes more or less identical with the input space-figure 74.

Additionally, the effectiveness and suitability of Lanczos approximation method in the case of kernel PCA is also demonstrated in the figure 74. Nystrom approximation technique selects a subset from a dataset in random way. That results in selecting a subset of the original dataset in a way that does not really represent sufficiently the original distribution then kernel PCA is performed. As one may observe in the figure 74 the output of Nystrom approximation is a reconstructed dataset that does not represent well the variance of the original dataset.

An important point here is that there is a trade-off between the number of selected principal components and the variance. In the case of KPCA it is desirable to obtain a restricted reconstructed dataset but also achieve the maximum variance. One may find in the example illustrated in the figure 74 that the default value of principal components (Number of Principal components=6) is a good trade-off. In the case of PCA the number of principal components is one since the given data set has originally two.

In statistics, the Bayesian information criteria (BIC) or Schwarz criterion is a criterion from model selection among a finite set of models. It is based, in part, on the likelihood function, and is closely related to Akaike criterion (AIC). The BIC is an asymptotic result derived under the assumption that the data distribution is in the exponential family and is described mathematically by the following formula, where n is the number of data points and k the number of parameters to be estimated:

$$\mathrm{BIC} = \chi^2 + k \cdot \ln(n).$$

Given any two estimated models, the model with the lower value of BIC is the one to be preferred. Lower BIC implies either fewer explanatory variables, better fit, or both. The before mentioned criteria may thus be considered as one approach to tune the number of components and hyper-parameters.

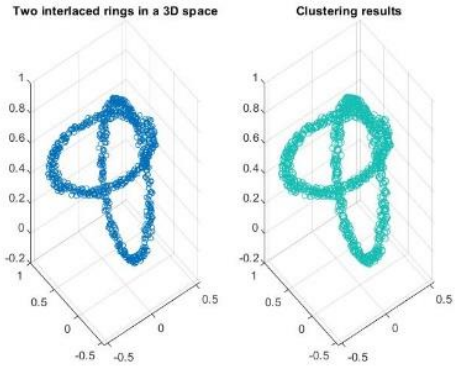
3.2 Spectral Clustering

Spectral clustering in the context of machine learning is a method that involves data grouping into categories based on some measure of inherent similarity. This is considered to be an instance of unsupervised learning. The corresponding supervised procedure is classification where a training set of correctly identified observations is available.

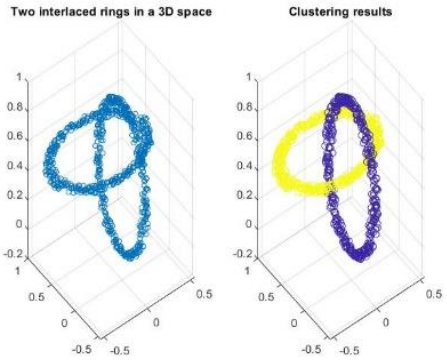
Spectral clustering works by first transforming the data from feature space into similarity space and then clustering in similarity space. The original data is projected into the new coordinate space which encodes information about how nearby data points are. The similarity transformation reduces the dimensionality of space and pre-clusters the data into orthogonal dimensions. This pre-clustering is non-linear and allows for arbitrarily connected non-convex geometries which is the main advantage of spectral clustering. This method is thus based only on the data points and needs no training set with prior information on the classes. Whereas in LS-SVM, needs a training set with prior information on the classes to learn, a validation set to tune the parameters and a test set to see the unbiased results.

If the method of spectral clustering is considered, one may observe that the kernel function acts as a similarity measure between two data points. By varying the values of σ^2 one affects the similarity criteria which results in categorizing the data points into different groups each time. This is illustrated in the examples presented in Figure 76 where an example of Spectral clustering for different values of σ^2 on an RBF kernel, takes place. In the case of σ^2= 0.001, the bandwidth of the RBF kernel is very small and as a result it is limited to measure very localized information. This might end up, to assign all points to the same cluster. Contrary to this, in the case of σ^2 = 0.2 or σ^2=0.05, the bandwidth of the RBF kernel becomes big, and as a result it ends up ignoring important local details and measure the global similarity which may result to mis-clustered data. In this case, σ^2=0.01 seems to be the best option.

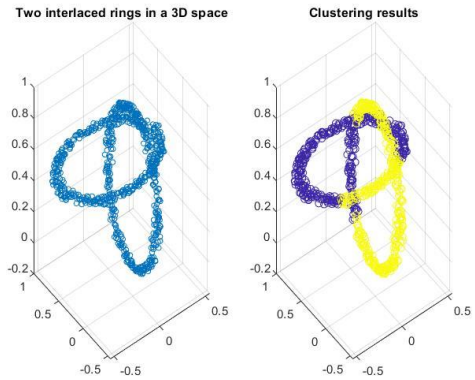
-Spectral clustering for various values of σ^2:



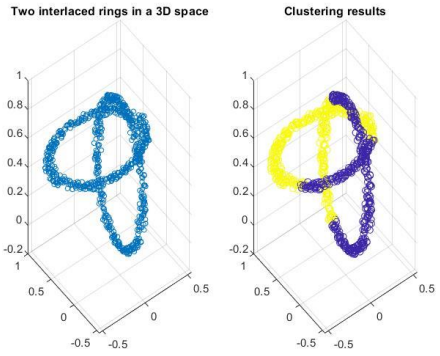
$\sigma^2 = 0.001$



$\sigma^2 = 0.01$

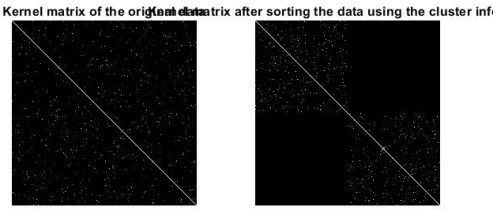


$\sigma^2 = 0.05$

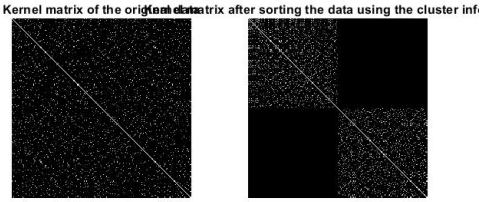


$\sigma^2 = 0.2$

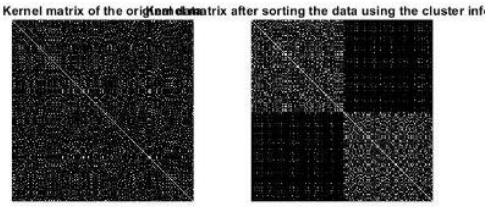
-Kernel matrix of the original dataset and after sorting the data using the cluster info:



$\sigma^2 = 0.001$

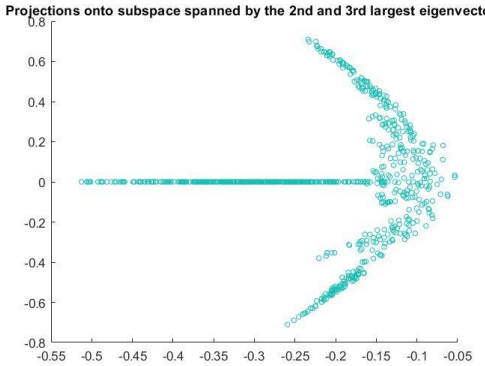


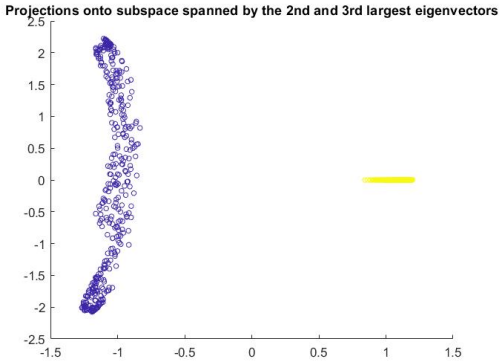
$\sigma^2 = 0.01$



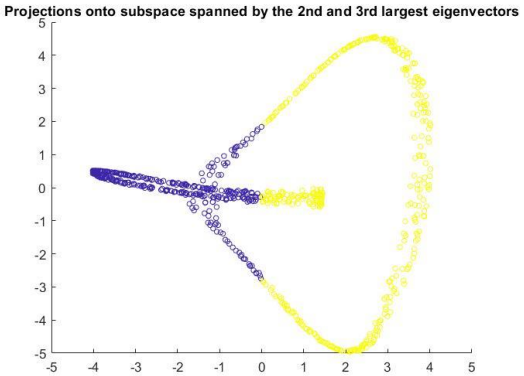
$\sigma^2 = 0.05$ $\sigma^2 = 0.2$

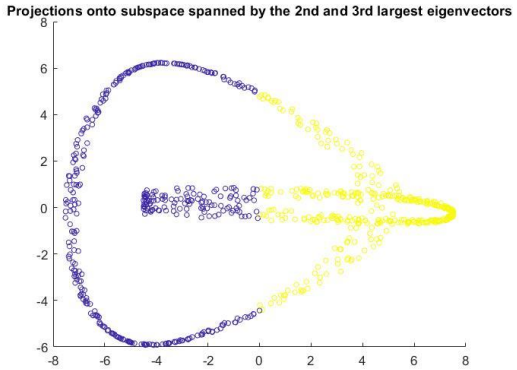
<u>Projections into subspace spanned by the 2nd and 3rd largest eigenvectors:</u>





$\sigma^2 = 0.001$ $\sigma^2 = 0.01$





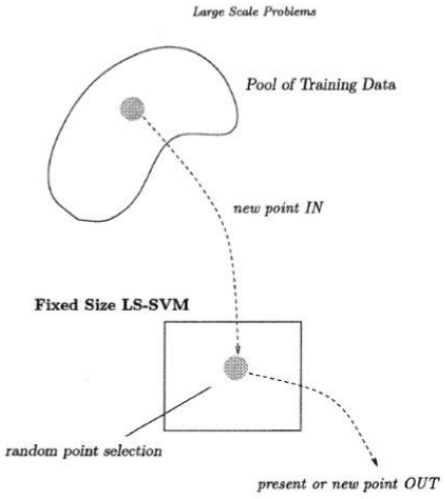
$\sigma^2 = 0.05$ $\sigma^2 = 0.2$

Figure 76 - Clustering for different values of σ^2 0.001, 0.01, 0.05 and 0.2 respectively

3.3 Fixed-size LS-SVM

Supposing the case of a linear kernel, someone can solve the primal problem as the dual problem. In fact, solving the primal problem is more advantageous for larger datasets while solving the dual problem is more suitable for large dimensional input spaces because the number of unknowns is equal to the number of training data points in this case. For example, in the linear function estimation case the mapping $\phi(.)$ becomes $\phi(x_k) = x_k$ and there is no need to solve the dual problem in the support values α, certainly not for large datasets. For the nonlinear case though, the situation is much more complicated. For many choices of the kernel, $\phi(.)$ may become infinite dimensional. However, one may still try in this case to find meaningful estimates for $\phi(x_k)$. A procedure to find such estimates is implicitly done by the Nystrom method, which is related to find a low rank approximation to the given kernel matrix by randomly choosing rows/columns of the kernel matrix.

Fixed size LS-SVM is a method for solving large scale regression and classification problems. In the Nystrom method, an approximate solution to the linear system is computed based upon a random subsample of the given training dataset. The number of support vectors is prefixed beforehand and the support vectors are actively selected from a pool of training data. After estimating eigenfunctions in relation the model is computed in the primal space with calculation of w, b. In the working set of support vectors, a point is randomly selected and replaced by a randomly selected point form the training data if this point improves the entropy criteria (related to density estimation and kernel PCA) then it gets selected. The LS-SVM model is estimated in the primal space thus one may consider fixed size-LS-SVM solves a linear problem. An intuitive description of the fixed size LS-SVM is shown in the following schematic.



Schematic 2: Fixed size LS-SVM

The reconstruction of the implicit feature space based on the Nystrom approximation induced by the kernel is formulated. In addition, the entropy criterion is introduced for the purpose of input selection. More particularly, sparse approximation of input samples can be related to the quadratic Renyi entropy

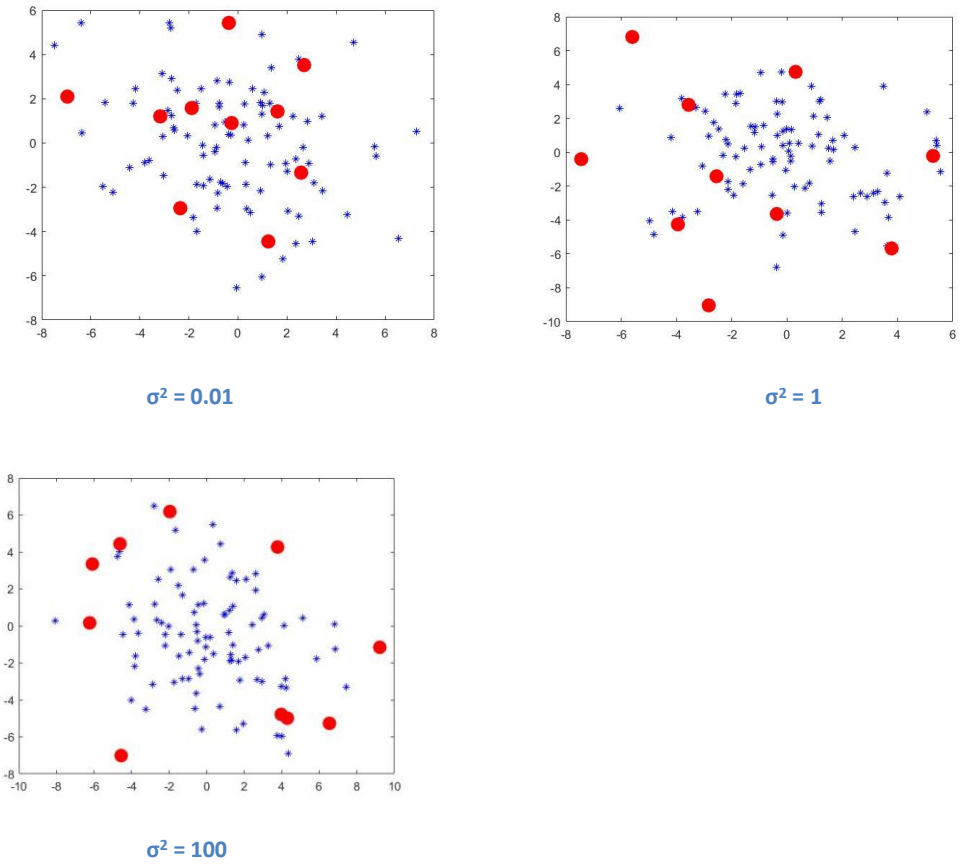
$$H_R = -\log \int p(x)^2 dx$$

which forms a measure of the distribution/compactness. An important remark is the choice of the kernel tuning parameter, namely the bandwidth σ of the RBF kernel. One may observe an interesting relation between the bandwidth of the RBF kernel and the scarcity/density of the feature space that is illustrated in figure 77. Model selection depends greatly on the choice of the bandwidth σ of the RBF kernel defined as

$$K(x_i,x_j) = \exp(-\|x_i - x_j\|_2^2/\sigma^2).$$

RBF kernel may be considered as a similarity function providing with pairwise similarity information. The relationship between two points can be interpreted differently for different values of the bandwidth by means of the entropy criteria. This means that the choice of the bandwidth controls the density of the generated subspace. When the selected value of the

bandwidth is small the density of the projected space is high whereas large values of the bandwidth result in sparse feature space formation-figure 77.



$\sigma^2 = 0.01$ $\sigma^2 = 1$

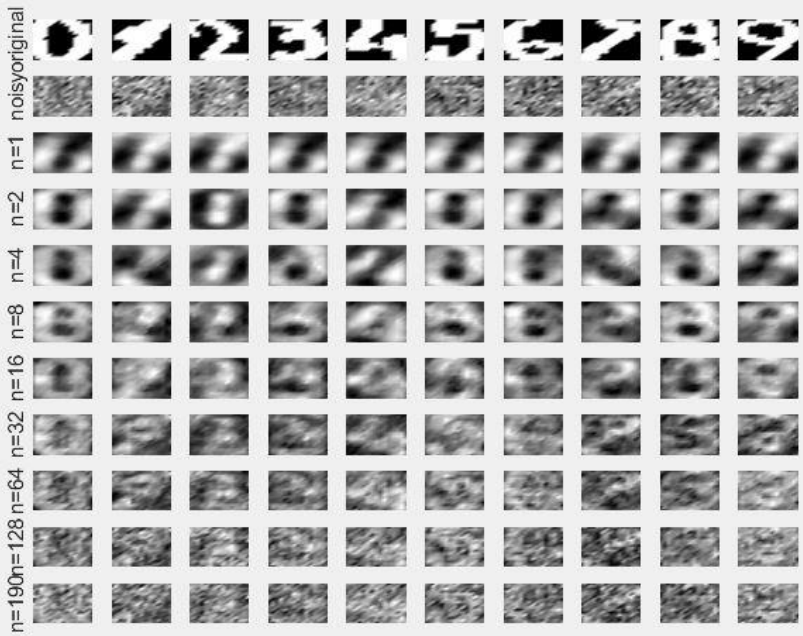
$\sigma^2 = 100$

Figure 77: Illustrates Nystrom approximation method for fix size LS-SVM using entropy criteria

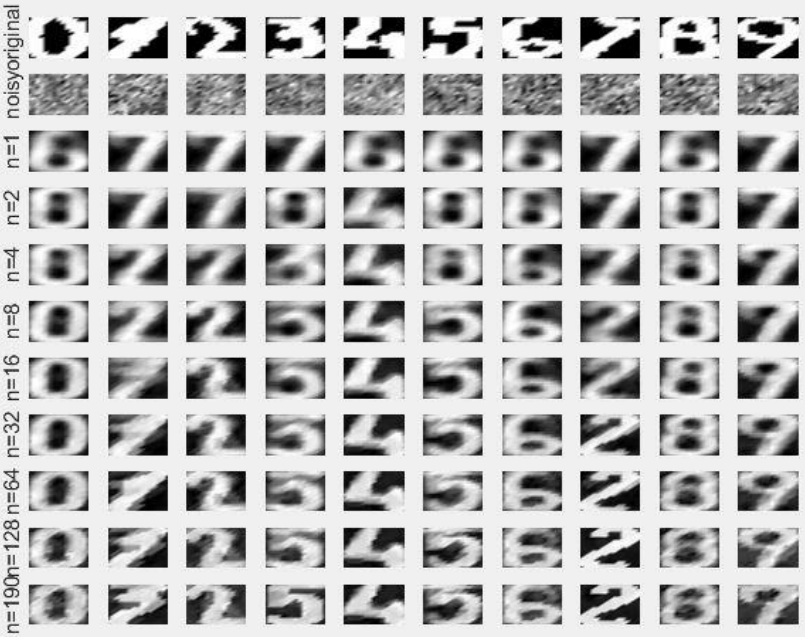
3.4 Homework problems

3.4.1 Kernel principal component analysis

The following figure (figure 78) illustrates the difference between linear and kernel PCA. It may be observed that kernel PCA performs better than classical PCA by means of denoising, although in this case where noisefactor is selected equal to 1.0, the performance of KPCA is moderate. More particularly, even in the case where the number of principal components selected is equal to 190 with PCA analysis is not possible to denoise patterns and obtain the original data set.



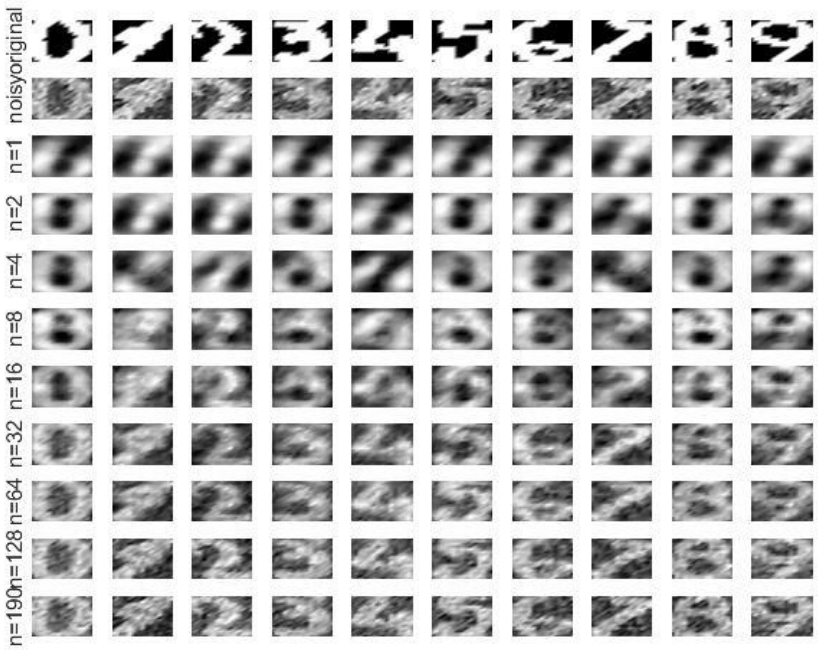
Linear PCA



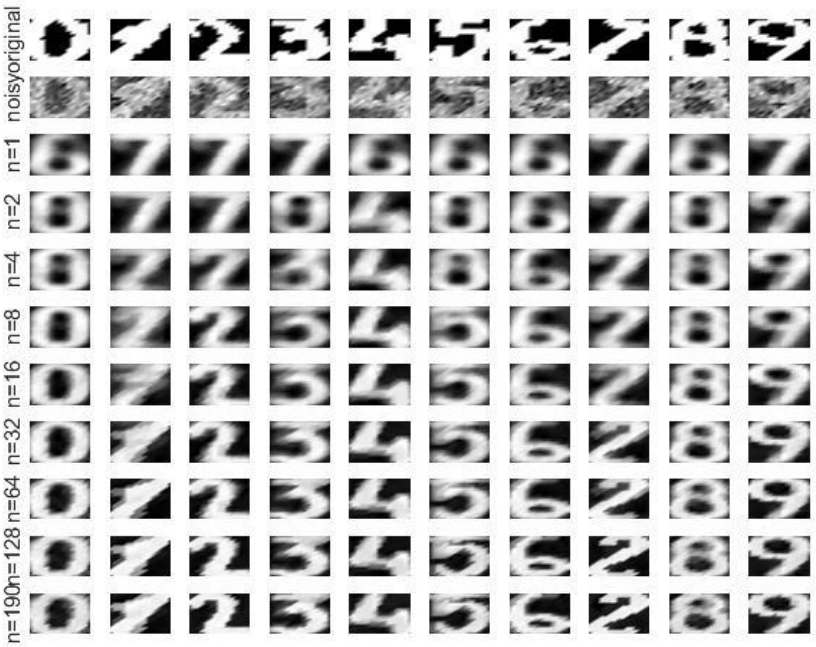
Kernel PCA

Figure 78: Illustrates denoising by means of (a) PCA (b) Kernel PCA using noisefactor=1.0

Furthermore, experiments prove that the selection of the sigmafactor parameter is crucial for the denoising process. The bandwidth σ is a parameter of the Gaussian kernel and can be used to balance between smoothness and accuracy. It may be observed that a variation of the sigmafactor which in turn means variations of the bandwidth, affects the denoising. For small values of sigma (Figure 79) KPCA method may reconstruct the original digits by using only one principal component, whereas by increasing the value of sigmafactor the performance drops and KPCA requires more principal components in order to denoise patterns sufficiently. It is also observed that in any case the kernel PCA outperforms PCA technique. However, one may observe that performance by means of denoising drops significantly even for KPCA in the case where the value of noise factor is set very high, as for example in the previous case where it is set equal to 1.0 (Fig 78).



Linear PCA



Kernel PCA

Figure 79: Illustrates denoising by means of (a) Linear PCA (b) Kernel PCA using noisefactor=0.3

Finally, in figure 80 we observe the reconstruction error (MSE) on training and validation sets, as a function of the kernel PCA denoising hyperparameter σ^2. As someone can see, for parameter value almost equal to 0.2, the validation errors for the test sets are minimal. Subsequently, using this optimized parameter in the denoising process, it is obvious that there is a significant improvement in the denoised results, apart from the digit number 5 which in the small number of components is still noisy (Figure 81).

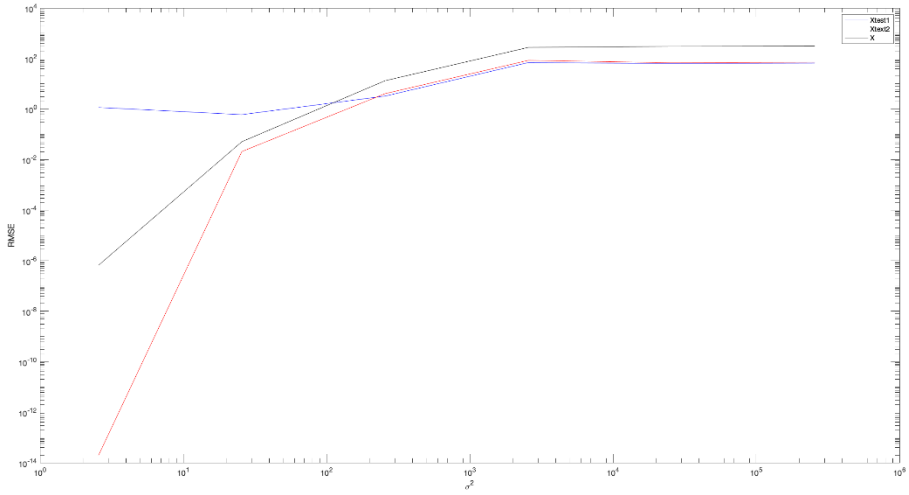


Figure 80: Illustrates the reconstruction error on training and validation sets as a function of σ^2.

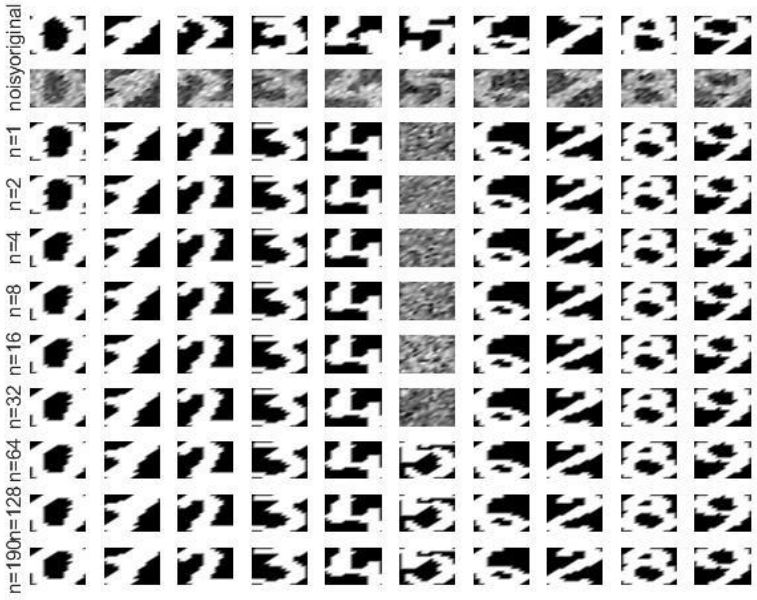


Figure 81: Denoising by Kernel PCA for $\sigma^2 = 0.2$.

3.4.2 Fixed-size LS-SVM

In this section, we investigate the use of fixed-size LS-SVM for two additional datasets: the Shuttle dataset (classification) and the California housing dataset (regression).

3.4.2.1 The Shuttle dataset

Subsequently, the methods of Fixed-size LS-SVM and l_0 - approximation are compared for the classification on the Shuttle dataset. The Shuttle dataset contains 58000 data points with 9 attributes all of which are numerical. Each data point is assigned to one of the 7 classes, where 80% of the data points are assigned to the first class. Since the dataset dimensionality is greater than two, it is difficult to visualize the dataset and as a result we do PCA and select the first two principal components.

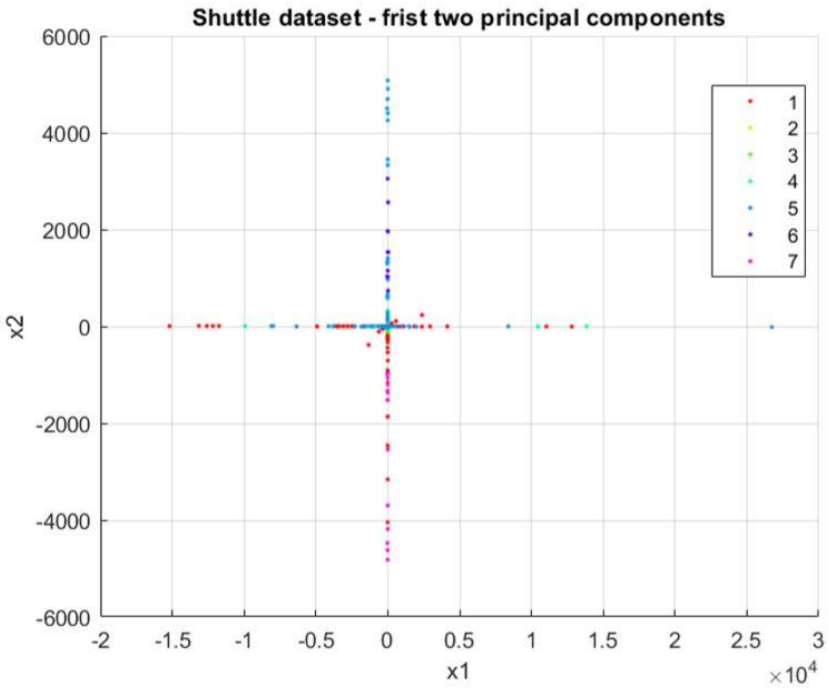
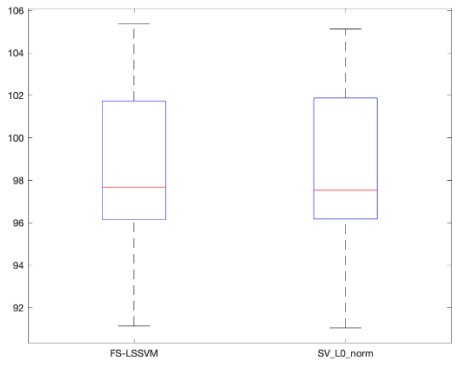
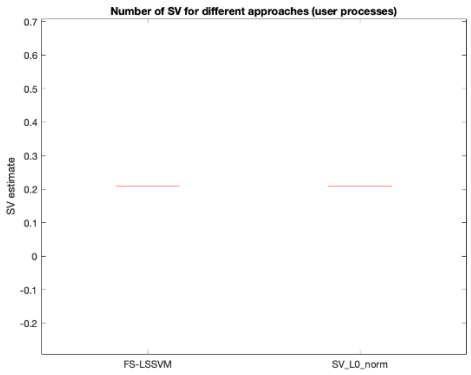


Figure 82: Shuttle dataset visualization

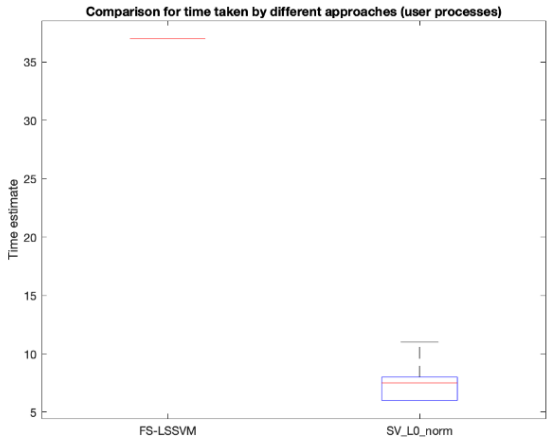
Since the computation time with RBF kernel is much higher, linear kernel is used in all the experiments. From the next figures we see that the distribution of error rates is nearly the same for both methods, however the fixed-size LSSVM has a higher median error rate. Also, the number of support vectors in the case of fixed size LSSVM used in constructing the decision boundary, is constant to the l_0 −norm solution, which at its lowest point uses only a fourth of the support vectors. Finally, the computation time is the same for both approaches.



(a)Error Comparison for the two approaches

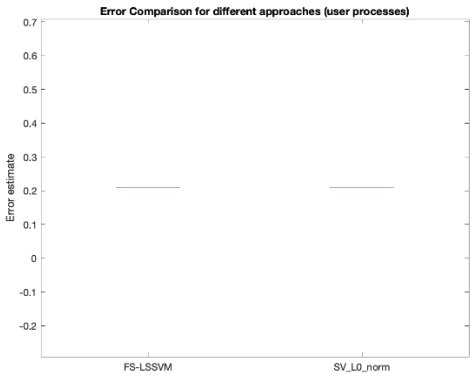


(b) Number of SV

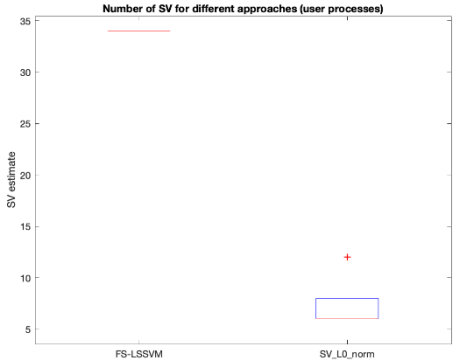


(c) Comparison of time taken by the two approaches

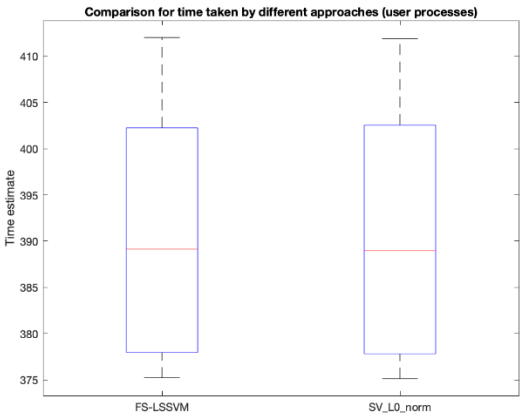
Figure 83: Illustrates the comparison of various metrics for FS-LSSVM and lo- approximation, using linear kernel for k=5.



(a)Error Comparison for the two approaches

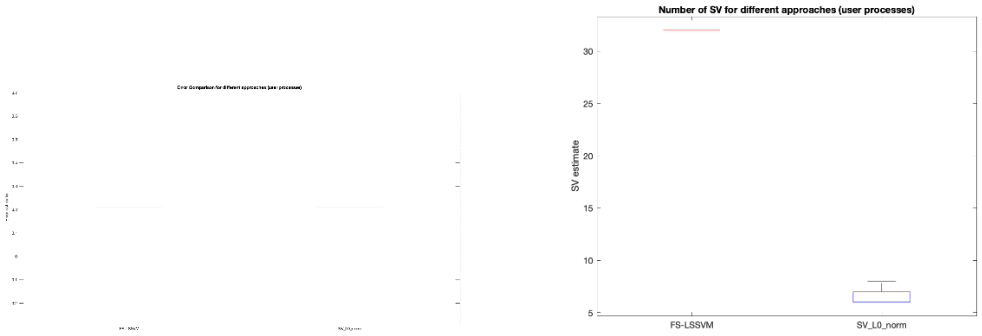


(b) Number of SV



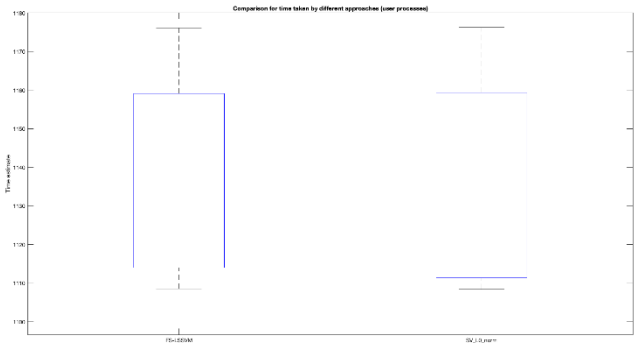
(c) Comparison of time taken by the two approaches

Figure 84: Illustrates the comparison of various metrics for FS-LSSVM and lo-approximation, using linear kernel for k=10.



(a)Error Comparison for the two approaches

(b) Number of SV



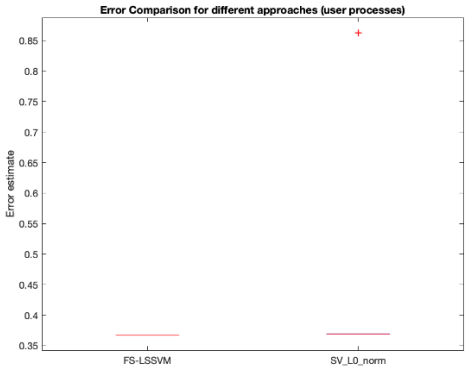
(c) Comparison of time taken by the two approaches

Figure 85: Illustrates the comparison of various metrics for FS-LSSVM and lo-approximation, using linear kernel for k=15.

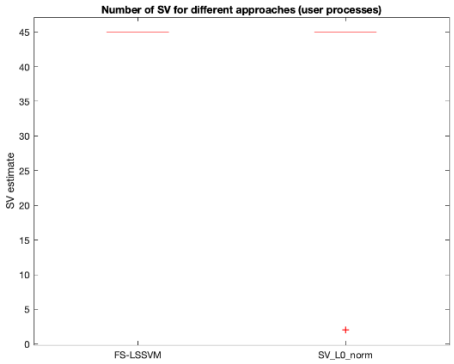
3.4.2.2. The California dataset

Finally, FS-LSSVM and l_0-approximation are compared in terms of error estimate, number of support vectors and computational time when using the California housing dataset for a regression task. The California dataset contains 20.640 data points and 8 attributes.

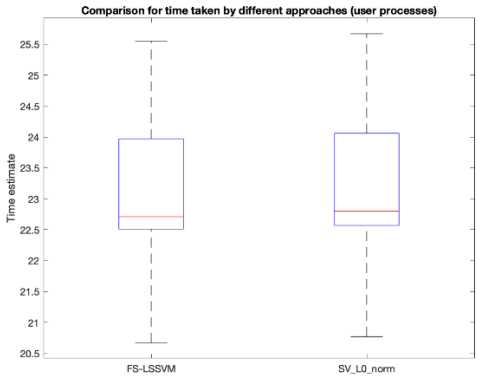
Figure 86 demonstrates the results of using a linear kernel. As someone can observe from the following figures, these two approaches perform similarly.



(a)Error Comparison for the two approaches



(b) Number of SV



(c) Comparison of time taken by the two approaches

Figure 86: Illustrates the comparison of various metrics for FS-LSSVM and lo-approximation, using linear kernel for k=3.

Lastly, Figure 87 demonstrates the results of using an RBF kernel. In this case the performance between the two approaches differs; the distribution of error rates is lower for FS-LSSVM, while the number of support vectors used in constructing the decision boundary

is higher in this case of FS-LSSVM. Concerning the computational speed, though, both methods perform similarly.

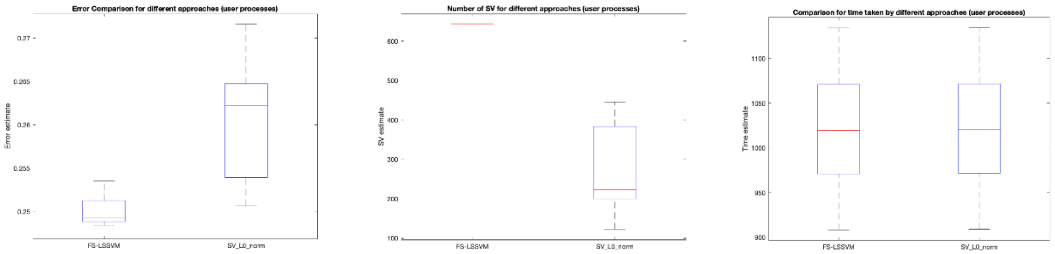


Figure 87: Illustrates the comparison of various metrics for FS-LSSVM and lo-approximation, using RBF kernel for k=5.

References:

---

1. Support Vector Machines: Methods and Applications, Johan A. K. Suykens
2. https://www.esat.kuleuven.be/sista/lssvmlab/tutorial/
3. https://medium.com/coinmonks/support-vector-machines-svm-b2b433419d73
4. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
5. https://www.youtube.com/watch?v=Z2_yh2sice8&feature=youtu.be&fbclid=IwAR2ZwLYOLyKg2p8fyV1PqOD46so67Hfdyjm8qMlngPN46dazjqoT1WoC4mg
6. https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72
7. https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496
8. https://archive.ics.uci.edu/ml/index.php
9. Center for Machine Learning and Intelligent Systems Statlog (Shuttle) Data Set.. Available online: https://archive.ics.uci.edu/ml/datasets/Statlog+(Shuttle)