Support Vector Machines

1. Exercise Session 1

1.1. A simple example: two Gaussians

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

For a binary classification problem, the Bayes classification rule leads to a minimal probability of misclassification. According to Bayes rule, a test observation x is assigned to a class i^* so that $i^* = arg\max_{i,...2} P(C_i|x)$, where $P(C_i|x)$ denotes the posterior class probability of class C_i. This also corresponds to minimizing the overlap area between the distributions $P(C_1|x)P(C_1)$ and $P(C_2|x)P(C_2)$. If the data are generated from a Gaussian distribution with the same covariance matrix for the two classes, the optimal decision boundary is a linear seperating hyperplane which is independent of the overlap area between these Gaussian distributions. As a consequence, in the case of non-seperable classes with overlapping distributions, misclassifications should be tolerated.

1.2. Support vector machine classifier

What do you observe when you add more data points to the dataset - both on the right and on the wrong side of the hyperplane. How does it affect the classification hyperplane?

Consider a given training set $\{x_k, y_k\}_{k=1}^N$ with input data $x_k \in \mathcal{R}^N$ and output data $y_k \in \mathcal{R}$ with class labels $y_k \in \{-1, +1\}$. According to Vapnik's extension of linear Support Vector Machines (SVMs) to the non-seperable case, the optimization problem in its primal form becomes:

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

subject to

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

where ξ_k slack variables measure the distance of each data point x_k from the marginal hyperplane. Slack variables allow a data point x_k example to be in the margin ($0 < \xi < 1$), or to be misclassified ($\xi > 1$). The term $c \sum_{k=1}^{N} \xi_k$ is used to penalize misclassification and margin errors. Additions on the wrong side of the hyperplane drastically change the decision boundary as they increase the value of the penalization term, as opposed to additions on the correct side of the hyperplane that have a minor impact to the solution. This behaviour is illustrated in Figure 1.

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

The regularization hyperparameter c determines how much emphasis is put on maximizing the margin versus tolerating misclassification errors. For a large value of c, a large penality is assigned to misclassification/margin errors. A smaller value for c, allows to ignore points close to the boundary, and increases the margin. As c is increased, the hyperplane's orientation is changed, providing a much smaller margin. For the case of a linear kernel, this effect is illustrated in Figure 2. Indeed, higher values of c produce a smaller margin and fewer support vector machines.

Kernel methods have been widely applied to support vector machines, as they allow to operate in a different feature space and construct non linear models. The optimization problem now becomes:

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

subject to

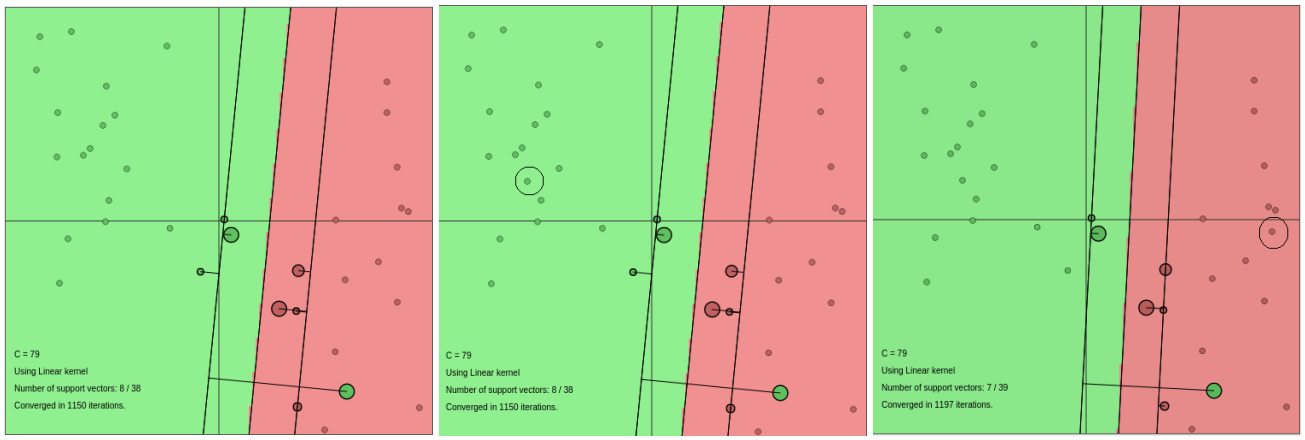
$$\begin{cases} y_k[w^T \phi(x_k) + b] \geq 1 - \xi_k, k = 1, ..., N \\ \xi_k \geq 0, k = 1, ...N \end{cases}$$

where $\phi(x_k)$ can be infinite dimensional. By constructing the lagragian, one can solve the problem in its dual form:

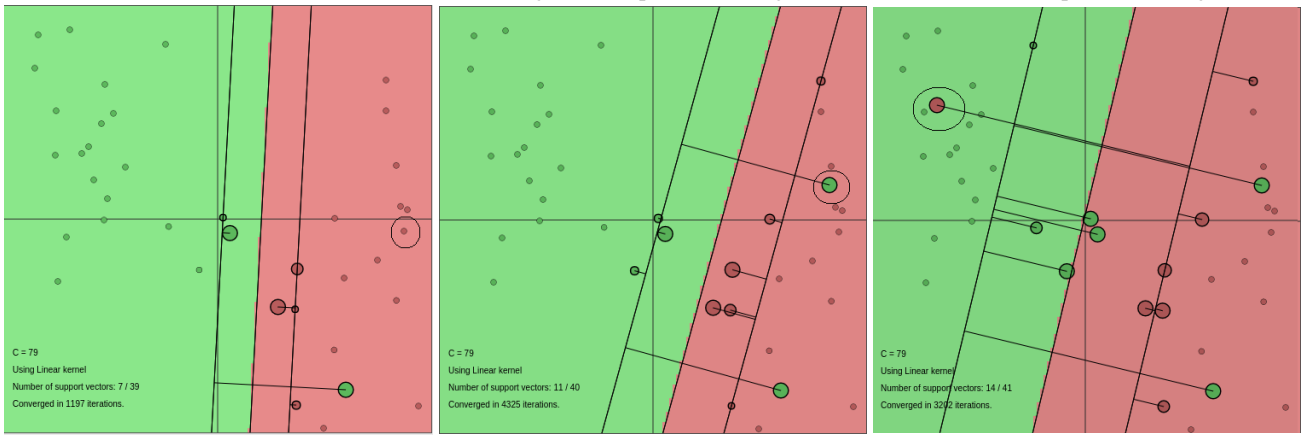
$$\max_{\alpha} J(w,b) = -\frac{1}{2} \sum_{k,l=1}^{N} y_k y_l K(x_k, x_l) \alpha_k \alpha_l + \sum_{k=1}^{N} \alpha_k \tag{3}$$

such that

$$\begin{cases} \sum_{k=1}^{N} \alpha_k y_k = 0 \\ 0 \geq a_k \geq c, k = 1, ...N \end{cases}$$

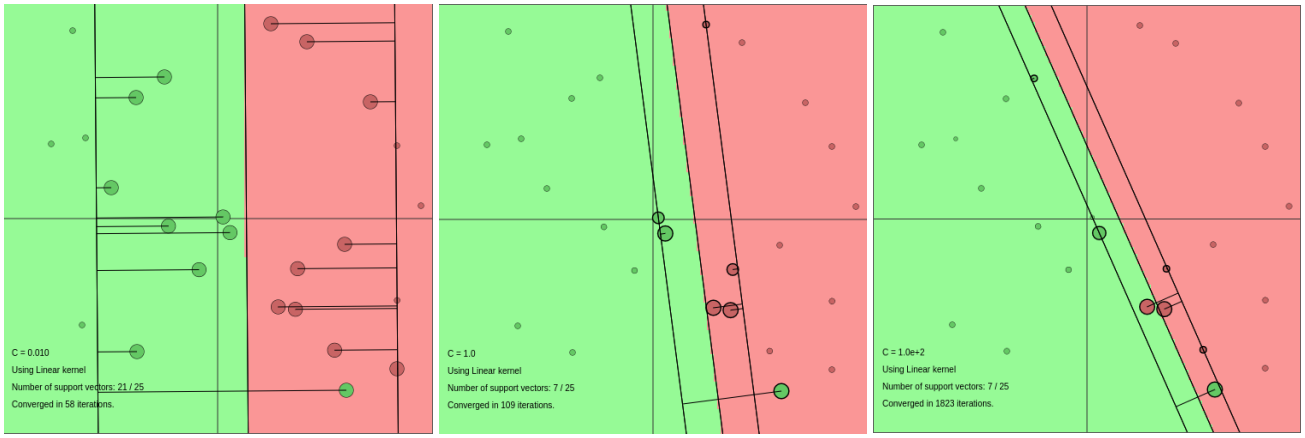


(a) original solution (b) new green data point on the right side (c) new red data point on the right side



(d) original solution (e) new green data point on the wrong side (f) new red data point on the wrong side

Figure 1. Adding more points to the dataset on the right side of the hyperplane (first row) and on the wrong side of the hyperplane (second row). Adding points on the wrong side of the hyperplane drastically changes the hyperplane, as opposed to adding points on the right side of the plane which has a minor effect.



(a) c=0.1 (b) c=1.0 (c) c=100

Figure 2. Varying the c parameter for the linear kernel. As c is increased from 0.1 (a) to 1.0 (b) and 100 (c) the hyperplane is adjusted and the margin becomes smaller as fewer misclassifications are tolerated.

where $K(x_k, x_l) = \phi(x_k)^T \phi(x_l)$ is the kernel function. The idea behind kernel methods is that if the data is not separable in the original space they can become separable in a higher dimensional space. This is better illustrated in Figure 3, for the case of 1-dimensional data. In the left plot, the original data points are shown which are not linearly separable in this 1-dimensional space. After applying the transformation $\phi(x) = x^2$ a second dimension is added to the feature space and the classes become linearly separable.

The kernel trick allows to work in feature spaces without actually needing to perform any computations in this space. The beauty of kernel methods lies exactly in their ability to represent the data only through a set of pairwise similarity comparisons between the original data observations (by using their original coordinates in the lower dimensional space), instead of explicitly applying the transformations $\phi(x)$. Finally, in order to predict the class of a new test instance x with a SVM classifier, a linear combination of kernel evaluations between the test instance and all the non-zero lagragian α variables should be computed:

$$y(x) = sign[\sum_{k=1}^{N} \alpha_k y_k K(x, x_k) + b] \tag{4}$$

We are particularly interested in the Gaussian (RBF) kernel:

$$K(x, x_k) = exp(-\frac{||x - x_k||_2^2}{\sigma^2}) \tag{5}$$

The Gaussian kernel measures the similarity between the test instance x and any training instance x_k as a function of the Euclidean distance. Since e^{-x} is a monotonically decreasing function, the higher the value of the term $\frac{||x-x_k||_2^2}{\sigma^2}$, the smaller $K(x, x_k)$ will be. If the distance between x and x_k is much larger than sigma, the kernel function tends to be closer to zero. With the use of Gaussian kernel, each training data point shapes a covering sphere around its center that subject to the width of σ. This effect is illustrated clearly in Figure 1.2 (e-g). Whenever the distance from a training instance exceeds this radius σ, the kernel value of this instance will get below the threshold value and its effect will vanish. Thus, if σ is very small, the data points will tend to be too dissimilar which can lead to over-fitting. In other words, smaller values for σ will tend to construct a classifier centered around local areas while larger values will produce the opposite effect. On the other hand, if σ is very large, the data points will tend to be very similar and under-fitting can arise.

Some observations are summarized below:

- For any given c, as $\sigma \to \infty$ the resulting classifier behaves like a linear classifier. Figure 1.2 (e-h) shows this effect as the classification boundaries become more and more linear as s is increased.

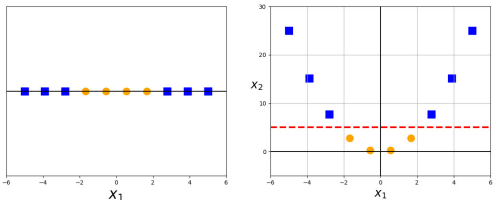


Figure 3. In the left plot, the original data points are shown which are not linearly separable in the 1-dimensional space. After applying the transformation $\phi(x) = x^2$ a second dimension is added to the feature space and the classes become linearly separable

- Under-fitting (all the data points are assigned the majority class) seems to occur in cases where σ is fixed and $c \to 0$ or $\sigma \to 0$ and c is fixed to a small value. Figure 1.2 (a) illustrates a case of underfitting with a small c and fixed σ.

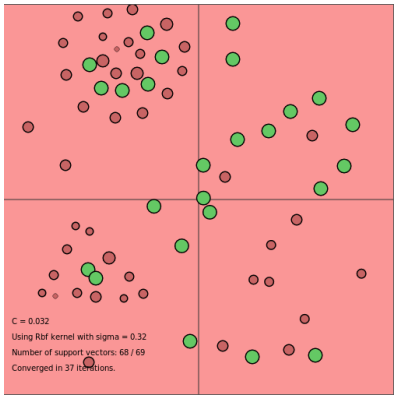
- Over-fitting (areas around the training data points of the minority class are classified as this class while the rest is classified as the majority class) occurs when $\sigma^2 \to 0$ and c is fixed to a large value. This case is illustrated in Figure 2 (e).

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

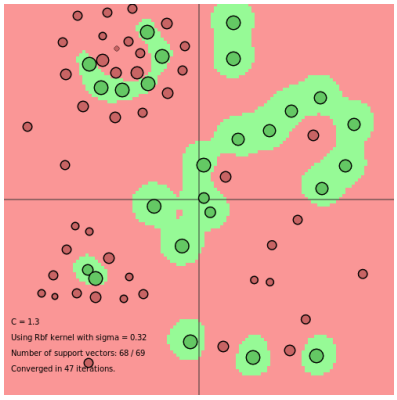
Figure 5 shows the result of applying a linear and a Gaussian kernel for the same set of training data points. Clearly, the Gaussian kernel performs better since it is able to produce non-linear decision boundaries. The linear kernel is not able to produce

What is a support vector? When does a particular datapoint become a support vector? When does the importance of the support vector change? Illustrate visually. Note that a support vector is indicated by a large circle with bold-lined circumference and that its importance is proportional to the size in the online application.

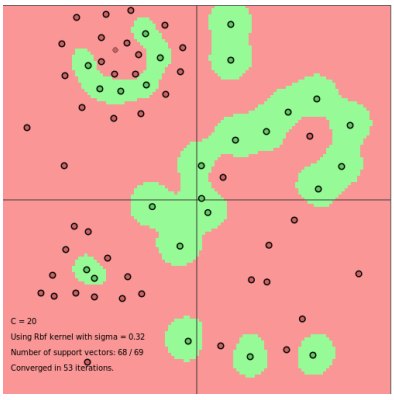
A property of the SVM is that the solution vector is sparse, which means that many of the resulting a_k values of equation (4) will be zero. The sum $\sum_{k=1}^{\#SV} \alpha_k y_k K(x, x_k)$ will therefore be computed only over the non zero a_k values instead of all the training data points. The training data points corresponding to non-zero a_k values are called support vectors and are located close to the decision boundary.



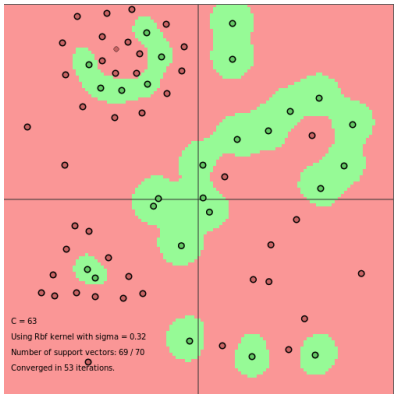
(a) $c = 0.032, s = 0.32$



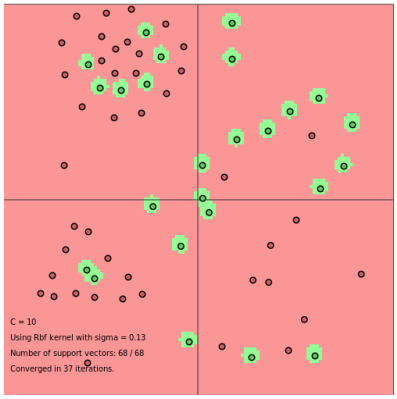
(b) $c = 1.3, s = 0.32$



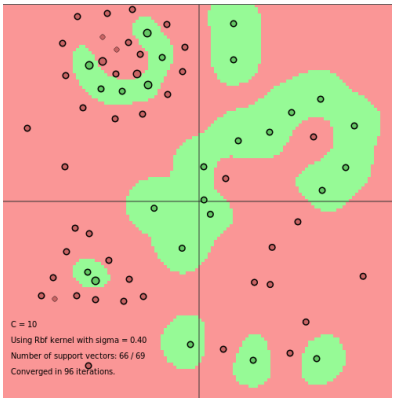
(c) $c = 20, s = 0.32$



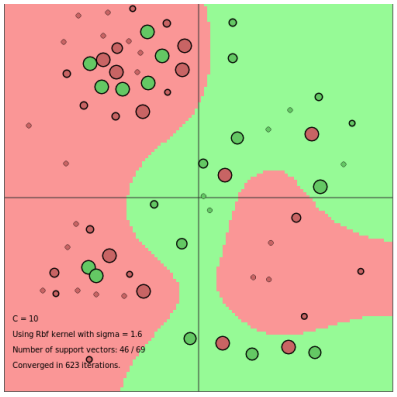
(d) $c = 63, s = 0.32$



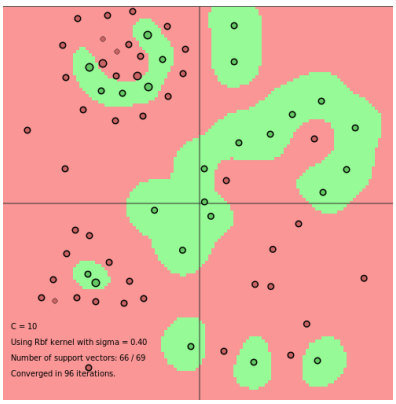
(e) $c = 10, s = 0.13$



(f) $c = 20, s = 0.43$

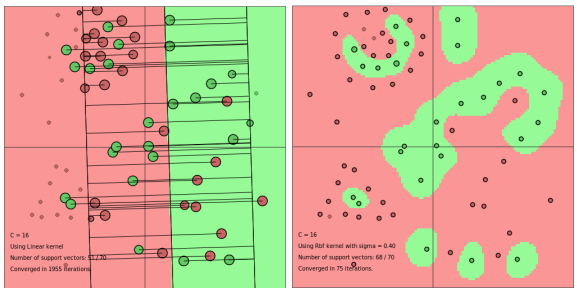


(g) $c = 10, s = 1.6$



(h) $c = 10, s = 20$

Figure 4. Varying c and σ for the RBF kernel.



(a) Linear kernel (b) RBF kernel

Figure 5. Comparison of the decision surfaces produced by a linear (a) and an RBF (b) kernel.

1.3. Least-squares support vector machine classifier

1.3.1. Influence of hyperparameters and kernel parameters

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

In this exercise, the objective is to classify the Iris dataset and explore the effect of using different kernel functions. The training set consists of a set of 100 data points $\{x_k, y_k\}_{k=1}^{100}$ with input data $x_k \in \mathcal{R}^2$ and output data $y_k \in \mathcal{R}$. The test set consists of 10 different data points.

For degree-d polynomials, the polynomial kernel is defined as:

$$K(x, x_k) = (x^T x_k + \tau)^d \tag{6}$$

where d defines the degree of the polynomial kernel. According to the Mercer theorem, the only condition for the Kernel function K is that it should be positive semi-definite. In that case, the convex nature of the dual SVM formulation problem guarantees that the solution found is going to be a global solution. The Mercer condition holds for all σ values in the RBF kernel as expressed in equation 5 and positive τ values in the polynomial kernel as described in equation 6.

As most practical questions in machine learning, the choice of the kernel parameters is data dependent. The typical procedure is to first train a linear kernel first and observe if performance is improved by using non linear kernels. The degree parameter controls the flexibility of the decision boundary. Higher degree kernels yield a more flexible decision boundary but also increase the risk of overfitting. As more and more parameters are added to the model, the complexity increases which in turn leads to models of low bias but high variance.

Figure 7 shows the decision boundary obtained for a linear kernel (a) and polynomial kernels of degree 2-4 (b-d). For the linear kernel, the classification error is 55% (e.g. 11 out of 20 observations are misclassified). Applying a quadratic kernel significantly improves the performance as the prediction error drops to 5%. For polynomials of degree ≥ 3 the prediction error drops to 0. In the case of high order polynomial kernels the decision boundaries become more and more complex which may lead to overfitting. This is illustrated in Figure 6 (e) and (f) where a polynomial kernel of degree 15 and 20 are applied. The prediction errors for the 15 degree and 20 cases are 5% and 20% respectively. To conclude, following the Occams razor principle, the polynomial with the lowest degree that performs best should be chosen. In this particular example, the ideal choice would be either a quadratic or a cubic polynomial since the prediction error is already very small or zero.

As more and more parameters are added to a model, the complexity of the model rises and variance becomes our primary concern while bias steadily falls

Lets now focus on the RBF kernel with squared kernel bandwidth σ^2.
-Try out a good range of different sig2 values as kernel parameters (fix gam = 1). Assess the performance on the test set. What is a good range for sig2?
-Fix a reasonable choice for the sig2 parameter and compare the performance using a range of gam. What is a good range for gam?

Values for parameters $\gamma =$ and σ^2 are varied from 0.01 to 100 and performance on the test set is assessed and presented in Figure 6. The optimal values are where the prediction error is close to 0. For $\sigma^2 \in [0.05, 5]$, the parameter γ can be set to $\gamma \in [0.5, 10]$ and achieve an accurate classification result with a prediction error of 0. Another possible range of values is $\sigma^2 \in [0.5, 5], \gamma \in [50, 100]$. As previously mentioned, a small error for γ allows to ignore points close to the boundary and increases the margin, resulting in tolerating more errors. An overestimated value value for σ raises the prediction error since the exponential in equation 5 will behave almost linearly and the higher-dimensional projection will lose its non-linear discriminative power. On the other hand, if underestimated, the decision boundary will become highly sensitive to noise as the model will start to overfit the data. In the next subsection, automated tuning algorithms for hyperparameter optimization are explored.

1.3.2. Tuning parameters using validation

Compute the performance for a range of gam and sig2 values (e.g., γ, σ^2 = 10 3 , . . . , 10 3). Use the random split method, 10-fold crossvalidation and leave-one-out validation. Visualize the results of each method: do you observe differences? Interpret the results: which values of gam and sig2 would you choose?

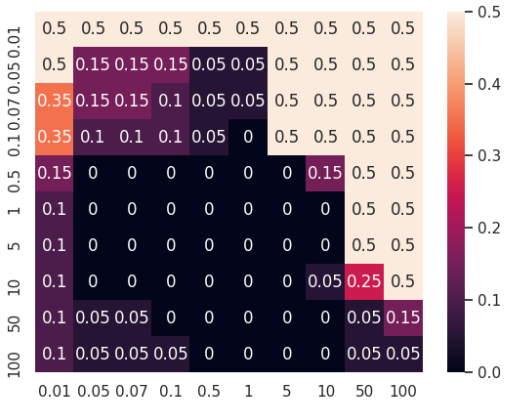


Figure 6. Prediction error on the test set is measured for a different range of values for σ (x-axis) and γ (y-axis). The figure suggests that a possible range of values for the two parameters is $\sigma \in [0.05, 5], \gamma \in [1, 10]$.

In the previous question, it was shown that the performance of the solution provided by a Least-squares support vector machine classifier depends on the choice of both the γ parameter and the parameters related to the kernel function (e.g. d in equation 6 and σ in equation 5). The goal of this exercise is to examine the efficiency and measure the performance of three different automatic tuning methods that split the raining data into a training and a validation set. Figure 9 shows the contour plots that describe the performance for each one of the three methods.

Random Split Random Split is the simplest approach to fine-tune different parameters. The training dataset is randomly split into a training and a validation set and the model is evaluated in the latter. Since the validation set includes data independent of that used for training, the performance of the validation set reflects the generalization ability of the model and the choice of different parameters can experimentally be tested. The validation set should not be confused with the test set which is meant to assess the performance of a fully-specified model. The error on the test set provides an unbiased estimate of the generalization error, assuming that the test set is representative of the population and follows the same probability distribution as the training set.

The main problem with the random split method is that it does not exploit the full set of data to train the model which may be problematic in cases where the size of the available data is small. The generalization error will depend on the split and different validation splits may give a different generalization errors. This means that the error estimates will be highly biased, resulting in a possibly large difference between the true test error and the estimated test error. The model will also has a high variance since it becomes very sensitive to the particular choice of data over which it is evaluated.

k-fold Cross Validation

In k-fold cross validation, the original data is randomly partitioned into k subsets of equal size. A single subset is used as a validation set and the model is trained on the remaining $k-1$ subsets. This process is repeated k times, with a different k subset used as a validation set each time. The obtained k results are averaged and a single estimation is produced to evaluate the performance.

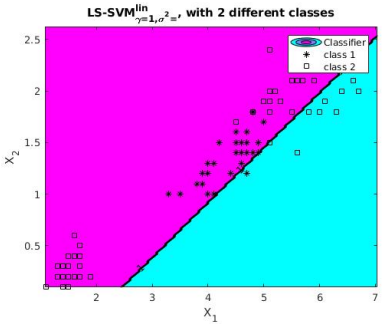
The k-fold cross validation method tries to overcome the bias induced by the random split method. If the mean performance is satisfying this means that the model gives low error on average, ensuring that the model's notions about the data should be accurate. On the other hand, examining the standard deviation of the errors can provide an insight about the model's variance. If the standard deviation of the errors is high, this means that the model's performance varies a lot with the different splits. The main advantage of this method mainly comes from using the whole training set, providing a statistical generalization of the the random split method.

The number of folds should allow the size of each of the validation sets to be large enough to accurately reflect the population. Some observations on the size of k are summarized below:

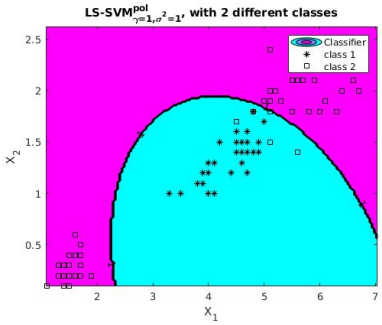
- A large value of k will result in a small bias of the error estimates and a higher variance.
- A small k will result in a high bias in the error estimates and in a lower variance

Leave-one-out validation Leave-one-out cross-validation is k-fold cross validation applied to its logical extreme, with the parameter k being equal to the number of available training instances N. The model is trained N separate times on all the available data except one observation which is used for test. The performance is estimated by averaging the errors and models are evaluated the same way as before.

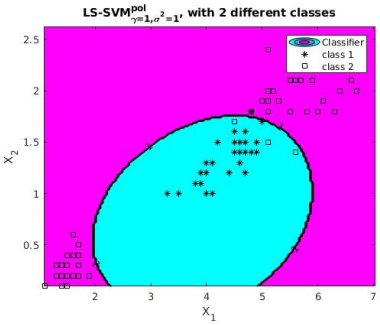
It is interesting to observe that the predictions made by the models on each leave-one-out fold will have less variability since the training set is almost the same each time (the full dataset minus one data point). Intuitively, the leave-one-out method provides has a lower bias compared to k-fold validation since almost all of the available data are used for training. The main drawback of this method is its computational complexity. For a training set that includes p examples, leave-one-out validation requires training the model p times which may become a problem for large datasets.



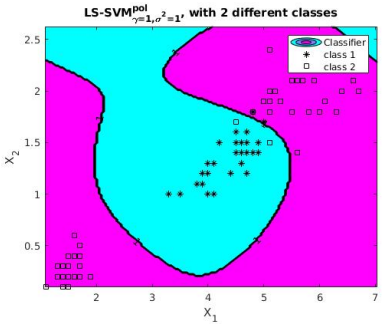
(a) Polynomial of degree 1. Prediction error: 55%



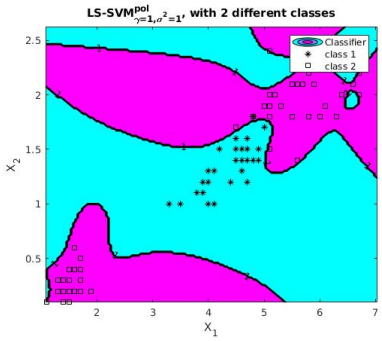
(b) degree 2. Prediction error: 5%



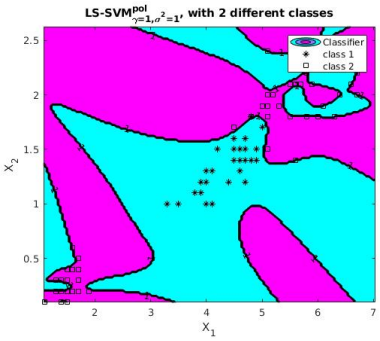
(c) degree 3. Prediction error: 0%



(d) degree 4. Prediction error: 0%

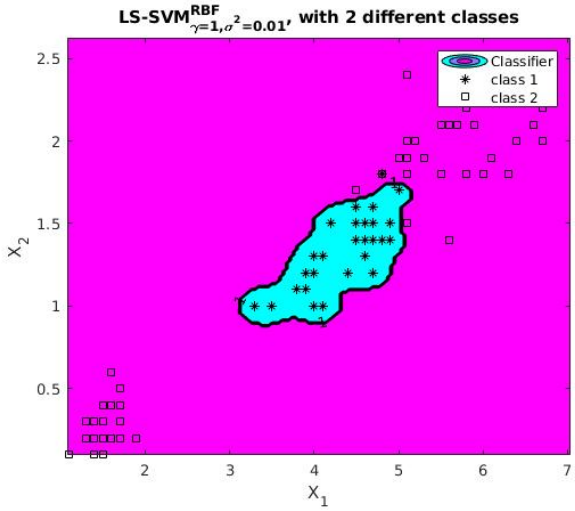


(e) degree 15. Prediction error: 5%

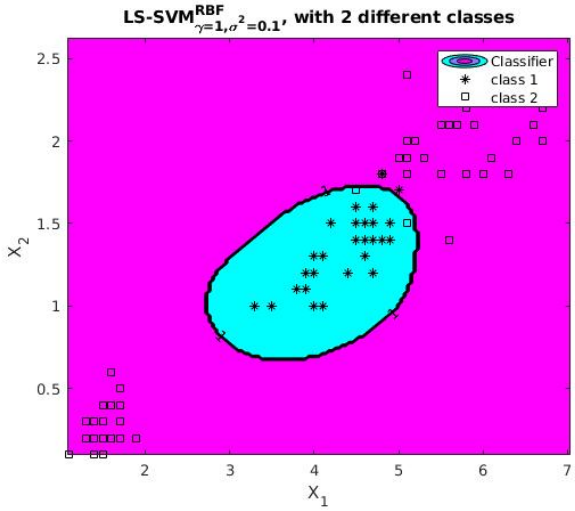


(f) degree 20. Prediction error: 20%

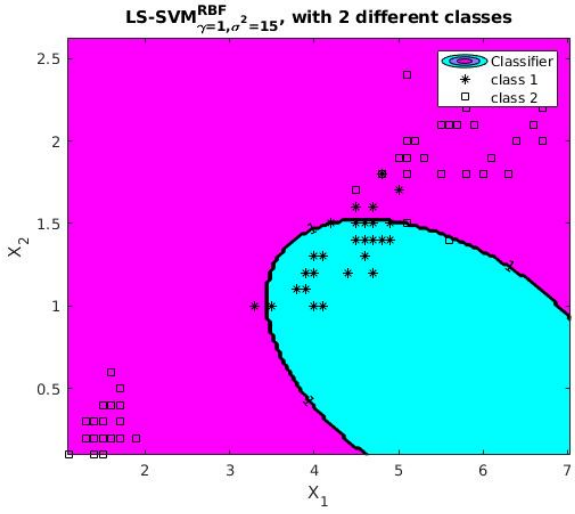
Figure 7. Varying the degree of the polynomial kernel. A linear kernel is unable to properly separate the data points. Increasing the degree of the polynomial kernel results in more expressive decision surfaces. However, if the degree of the polynomial kernel is too high this results in overfitting.



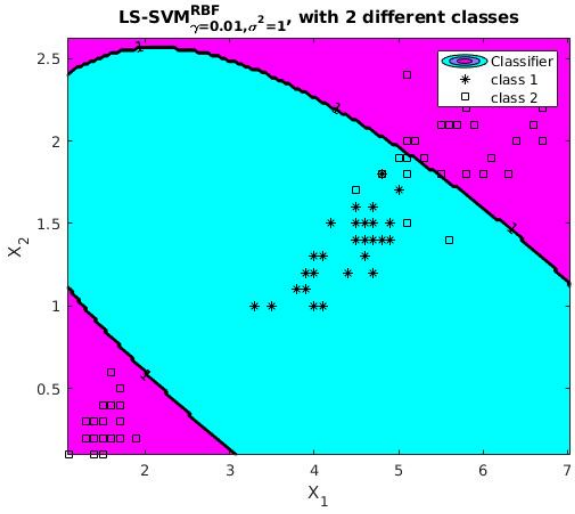
(a) $\gamma = 1, \sigma = 0.01$



(b) $\gamma = 1, \sigma = 0.1$

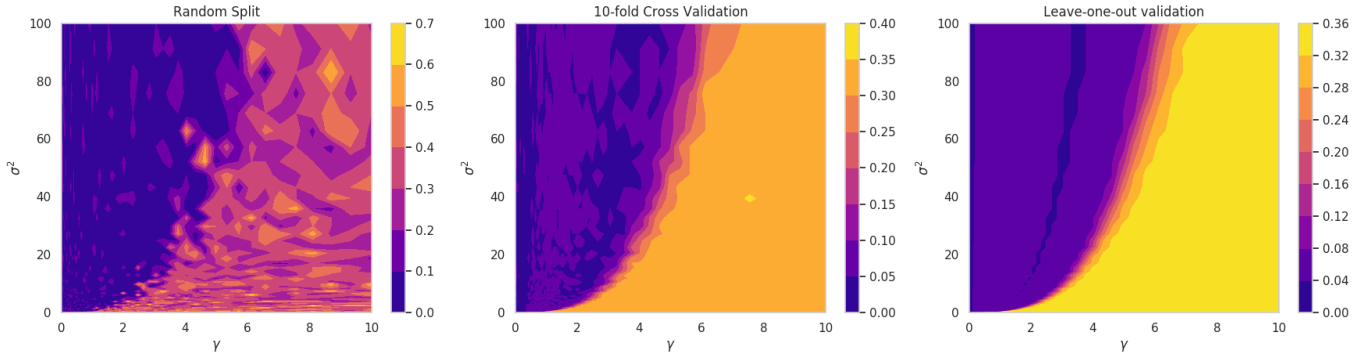


(c) $\gamma = 1, \sigma = 15$



(d) $\gamma = 0.01, \sigma = 1$

Figure 8. Varying σ and γ parameters in the RBF kernel. In (a) the decision boundary is centered around the points of one class which impacts the generalization ability of the classifier. In (d)



(a) Random split (b) 10-fold cross validation (c) leave-one-out cross validation

Figure 9. Visualization of the of misclassification error as a function of parameters σ^2 and γ for Random split, 10-fold validation and leave-one-out validation

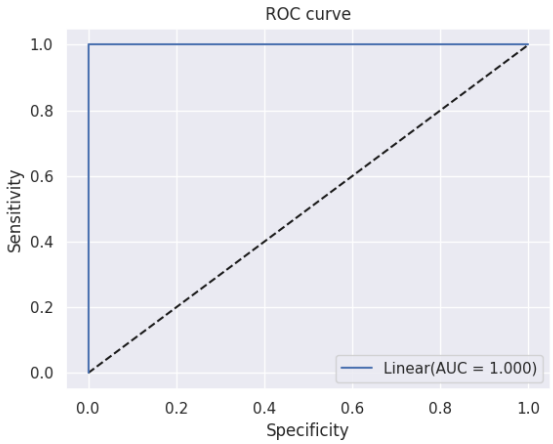


Figure 10. ROC curve for the test partition of the iris dataset

1.3.3. AUTOMATIC PARAMETER TUNING

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

Direct Search methods are optimization methods that do not use derivatives approximations. Instead, they sample the objective function at a finite number of points in each iteration and decide on which actions to perform based on these values. In this section, two different approaches for automatic parameter tuning are tested and compared: NelderMead method and a grid search method. The Nelder-Mead method is a heuristic search method that belongs to direct search optimization techniques and is often applied to nonlinear problems. The grid search works by evaluating all the different values for the parameters to be tuned within a certain range. Table 1 summarizes the tuning results. The obtained hyperparameters differ every time the tuning is performed since the objective function is not convex and multiple minimum solutions may exist.

First, the suitable starting points are determined using Coupled Simulated Annealing (CSA) and then the starting points are passed to the selected optimization algorithm.

1.3.4. ROC CURVES

In practice, we compute the ROC curve on the test set, rather than on the training set. Why? Generate the ROC curve for the iris.mat dataset (use tuned gam and sig2 values).Interpret the result

The receiver operating characteristic (ROC) curves can be used to express the relationship between the sensitivity and the specificity of a classifier along different thresholds. The sensitivity (or true positive rate), plotted on the y-axis, can be described as:

$$TPR = \frac{TP}{TP + FN} \tag{7}$$

where TP and FN (true positive and false negative) refer to the number of correctly and incorrectly classified positive instances. Therefore, sensitivity describes the fraction of correctly classified instances over the total number of positive examples. On the other hand, specificity (or false positive rate) is computed as:

$$FPR = \frac{TN}{TN + FP} \tag{8}$$

where TN and FP (true negatives) stand for the number correctly and incorrectly classified negative instances. Specificity refers to how accurate the classifier identifies negative cases. By iteratively setting the threshold of the classifier, the curve is produced. Naturally, The area under the curve (AUC) of an ROC plot describes the quality of the classification model. If the AUC score is 1, this means that the model is able to perfectly separate the data. On the other hand, if the area equals 0.5 the classifier has no discriminative power, it simply behaves as a random classifier. The ROC curve is simply computed over the test (or validation set) and not the training set since we are interested in examining the generalization ability of the classifier. As discussed in section 1.3.2, the performance on the test set is used to provide an unbiased evaluation of the final model.

After tuning the hyperparameters c and σ, the ROC curve is plotted and shown in Figure **??**. The result shows that is the hyperparameters are properly selected, the obtained classifier can perfectly separate the data, having an AUC score of 1, a FPR of 0 and a TPR of 1. All of the hyperparameters described in Table 1 produce the same result.

1.3.5. BAYESIAN FRAMEWORK

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

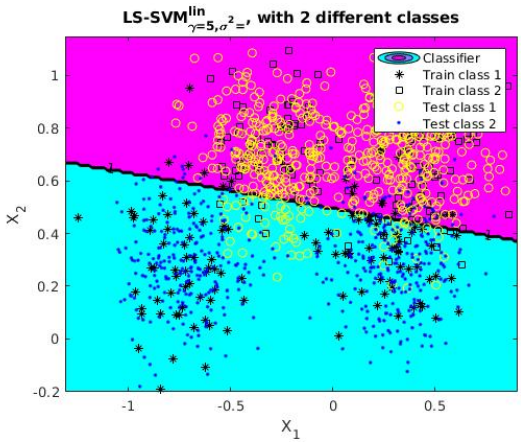
1.4. Homework Problems

1.4.1. CLASSIFICATION OF THE RIPLEY DATASET

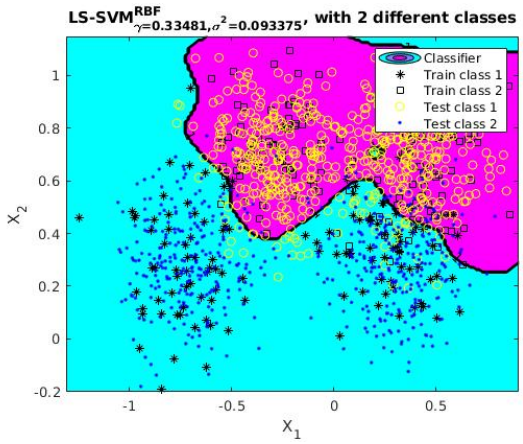
The Ripley dataset consists of a set of input data $x_k \in \mathcal{R}^2$ and output data $y_k \in \mathcal{R}$ with class labels $y_k \in \{-1, +1\}$, with 250 data points intended for training and 1000 points intended for test. As illustrated in Figure 11 (d), the two classes seem well separated from each other, although not entirely linearly separable. In the same Figure the results of applying the linear (a) and the RBF (b) kernel are also shown. The linear kernel achieves a classification accuracy

	Nelder-Mead method			Brute Force Gridsearch		
	γ	σ^2	cost%	γ	σ^2	cost%
CSA						
	6.0623	1.2143	0.0400	0.1930	5.49	0.04
RDS						
	1.4513	5.7490	0.04	0.0674	0.57	0.03

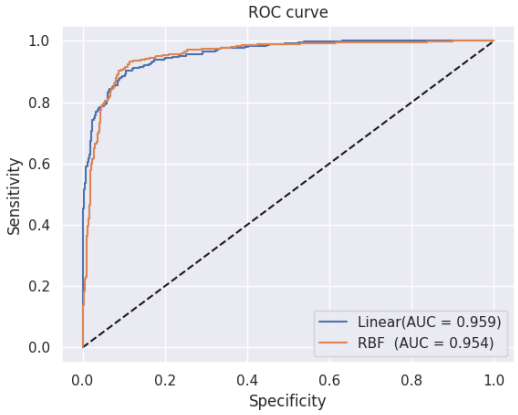
Table 1. Hyperparameters tuned with Nelder-Mead method and Brute Force Gridsearch



(a) Linear kernel



(b) RBF kernel



(c) Linear kernel

(d) The Ripley dataset

Figure 11. Classifying the Ripley dataset

of 0.89, while the RBF kernel produces a non-linear decision boundary and reaches an accuracy score of 0.90. For the RBF kernel, the hyperparamteres c and γ were tuned using the Nelder-Mead method, as described in section 1.1.6. By plotting the ROC curves in Figure 11 (c), it is is clear that the performance of both kernels is very similar: an AUC score of 0.959 is achieved in the linear case and 0.954 in the RBF case. The conclusion is that any of the two kernels can achieve a good classification performance but since we always opt for the simplest solution the linear kernel is preferred.

1.4.2. Classification of the The Breast Cancer dataset

The Breast cancer dataset consists of 569 examples, with 400 intended for training and 100 for testing. The data points are composed of 30 features that describe metrics suitable for classifying cancer types as benign or malignant. The Bengin class accounts for 62.74% of the diagnosis class while the malignant class accounts for 37.26%. The test set follows the same class distribution as 107 out of 169 instances belong to the negative class and 69 to the positive. In Figure 12 the distributions of the values of the features are shown. Each plot corresponds to one feature. It is observed that the overlap between the two classes is smaller for some of the features (e.g. concave points worst, area worst, concave mean). Moreover, in some cases the values for class -1 are significantly larger (e.g. area se, perimeter se, readius mean, etc). Last but not least, it is shown that in some cases the variance of feature values are remarkable different (e.g. radius worst).

It will be interesting to explore whether we can reduce the dimensionality of the input data points by performing Principal component analysis (PCA). PCA is a dimensionality reduction technique that can be used to project the data observations into a lower dimensional space while maximizing the variance of the projected data. In short,after standardizing the input vectors, the Covariance matrix $C = \{xx^T\}$ that contains all the possible covariance pairs of the initial variables is computed. This is because the covariance matrix expresses how the variables of the observations are varying from the mean with respect to each other. The eigenvectors of this matrix show the directions where the variance is maximal and are called principal components. The amount of variance each principal component carries is expressed by the eigenvalues which can be used to rank the eigenvectors in order of significance.

While PCA seems a promising solution to reduce the dimensionality while keeping the relevant information, it should be noted that it does not guarantee to always produce a better classification result. This is because the direction of maximal variance does not necessarily make the input data more separable. However, in a lot of cases applying PCA before training can improve the performance or simply allow to reduce the dimension of the input vector. Finally, reducing the dimension of the input vectors can be particularly useful when solving the SVM problem in its primal form where the complexity depends on the dimensionality of the data.

	Linear	RBF
2	0.9172	0.9290
5	0.9527	0.9645
8	**0.9645**	**0.9882**
10	0.9586	0.9467
15	0.9586	0.9586
30	0.9527	0.9467

Table 2. Classification accuracy on the Breast cancer dataset after projecting the input data to 2-30 principal components

After scaling the data by subtracting the mean observation and by dividing with the standard deviation, a LS-SVM classifier with a linear kernel is trained by using all of the 30 features. The input data is also projected to a different number of principal components and the results and each time are compared. Table 2 summarizes the results. In the case where all of the 30 features are used, the classifier achieves an accuracy score of 0.9527. It is also shown that by using the first 8 principal components, the classification accuracy is improved from 0.9527 to 0.9645. Interestingly enough, by using only 5 features instead of 30 the result remains the same. The same experiment is repeated for the RBF kernel. First, γ and σ^2 are tuned using the Nelder-Mead method and then the LS-SVM model is trained by using input vectors projected into a different number of principal components. For $\gamma = 41.365$ and $\sigma^2 = 44.38$ and by prjojecting the input evctors to 8 principal components, the best classification score of 0.9882 is achieved. Finally, plotting the ROC curves for both kernels shows that the RBF kernel reaches a very good AUC score of 0.996, while the same score for the linear kernel is at 0.952. To conclude, in this case by performing PCA the relevant information about the features seems to be contained in the projected data and the classification accuracy is improved both in the linear and the RBF case.

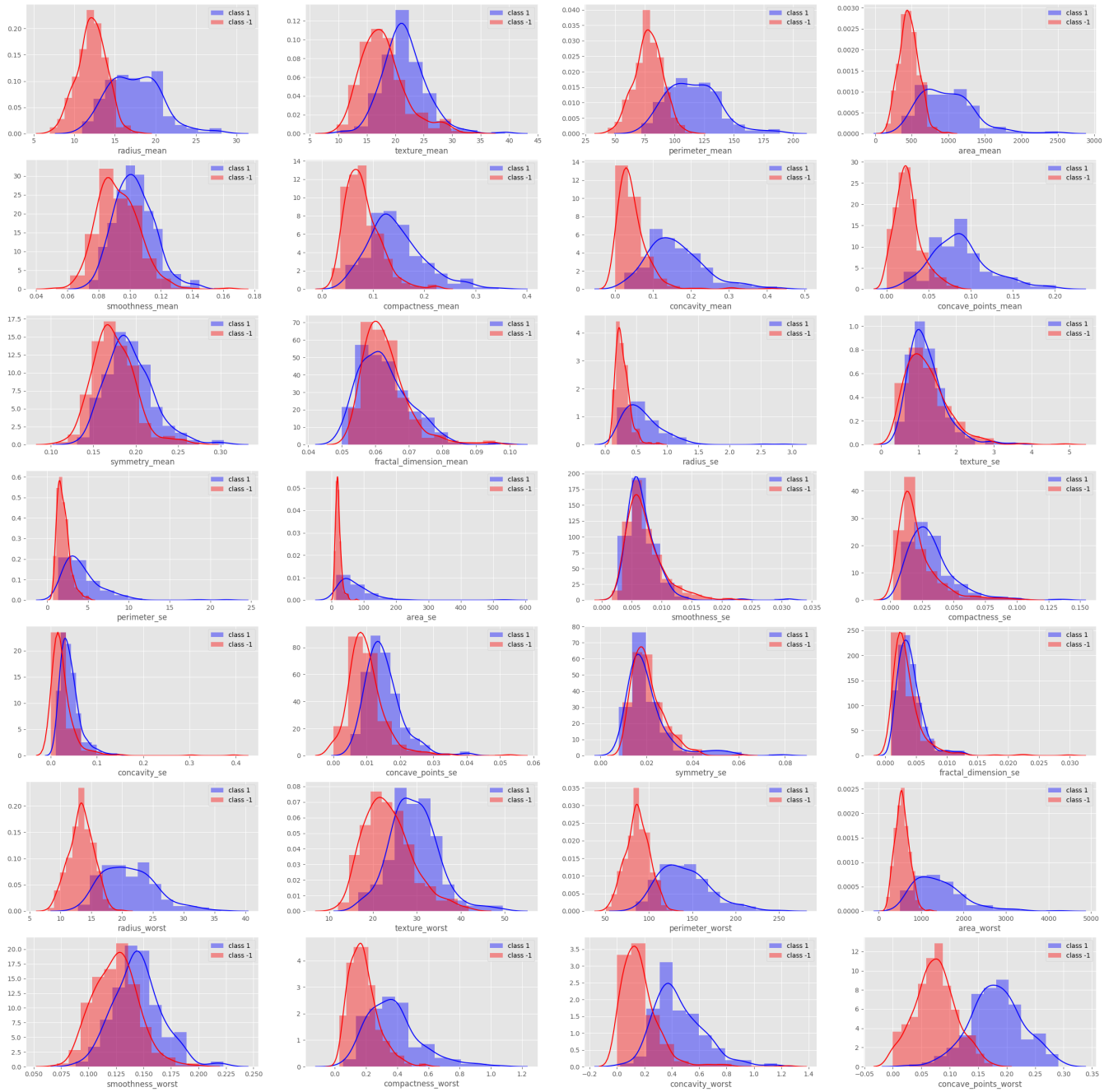
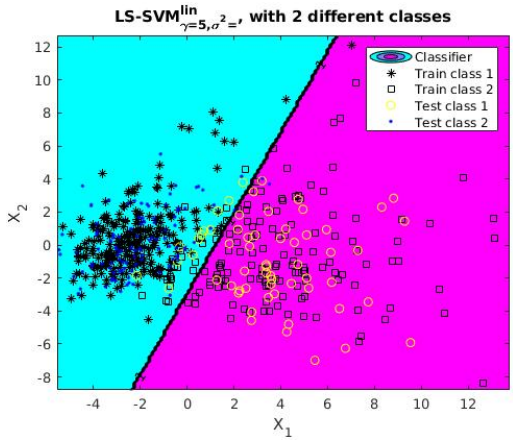
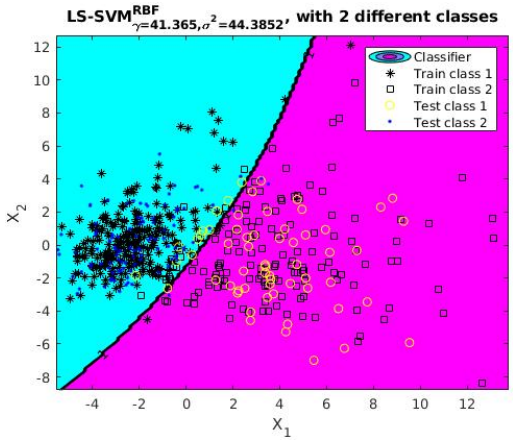


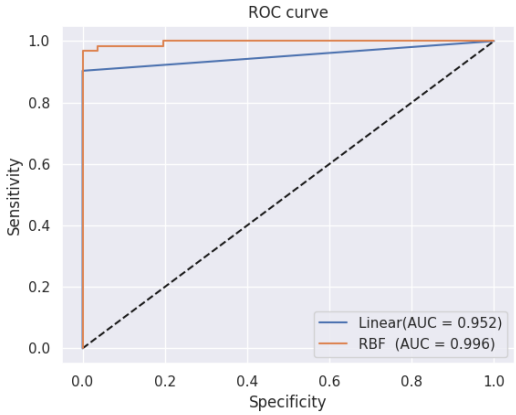
Figure 12. Distribution of features for the Breast cancer dataset



(a) Linear kernel



(b) RBF kernel



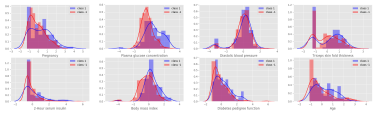
(c) Linear kernel

Figure 13. Classifying the Breast cancer dataset

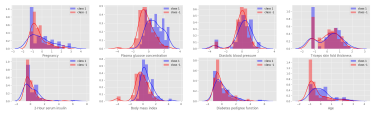
1.5. Classification of the Diabetes dataset

The diabetes dataset consists of 468 instances, with 300 intended for training and 168 for testing. The data points are composed of 8 features: Plasma glucose concentration, Diastolic blood pressure, Triceps skin fold thickness, 2-Hour serum insulin, Body mass index, Diabetes pedigree function, Age and finally the number of times the patient was pregnant. Figure 14 shows the distribution of the two classes over the 8 different features. The first thing to note is that there seems to be a high overlap between the distributions of the two classes. Secondly, the feature values of the test partition seem to follow the same distribution as the values in the train partition.

In this exercise, the same methodology as before is followed: First the data points are scaled by subtracting the mean observation and dividing with the standard deviation. Then, a LS-SVM classifier is trained by using all of the 8 features. Next, the input vectors are also projected to 2-8 principal components and results are reported in terms of classification accuracy and ROC curves. Table **??** summarizes the outcome of this experiment. It is shown that in this case PCA does not produce a better classification result, since the best accuracy score is obtained by training with all of the 8 features. For the RBF kernel the classification accuracy is 0.8036 while for the linear case the accuracy result is 0.7440. Figure **??** shows the ROC curves for the linear and the RBF kernel. The linear kernel reaches an AUC score of 0.670, while the RBF kernel performs better having an AUC score of 0.844. To conclude, the result is not as satisfying as with the previous two datasets but this is mainly attributed to the fact that there is a high overlap between the values of the available features.



(a) train set



(b) test set

Figure 14. Distribution of features for the diabetes dataset

2. Exercise Session 2