Instituto Superior Técnico

Departamento de Engenharia Electrotécnica e de Computadores

Machine Learning

5^{th} Lab Assignment

Shift *Sexta 14h* Group number *1*

Number *84053* Name *Francisco Raposo de Melo*

Number *89213* Name *Rodrigo Tavares Rego*

Support Vector Machines for Classification

1 Introduction

Simple linear classifiers, such as the one implemented by the Rosenblatt perceptron, are unable to correctly classify patterns, unless the classes under consideration are linearly separable. Neural networks that use hidden units with nonlinear activation functions are used in many classification problems, since they are able to perform nonlinear classification. However, several strong theoretical results, valid for the linearly separable case, are not applicable to nonlinear classifiers.

Support vector machines (SVMs) address the classification problem using linearly separable classes, not in the input space, but in the so-called *feature space*. Input patterns are mapped onto the higher-dimensional feature space, where the classification is performed using a hyperplane as classification border. Since the mapping from the input space to the feature space is usually nonlinear, these hyperplanes in feature space correspond to nonlinear borders in input space.

At first glance this might seem to be a double-edged sword, since it suggests that calculations have to be performed in the high-dimensional feature space. However, an interesting result proves that, since linear classification only requires inner product operations, all calculations can be performed in the lower-dimensional input space, if the nonlinear mapping is chosen in an appropriate way. This result is particularly strong when one takes into account that certain mappings yield infinite-dimensional feature spaces. This is the same as saying that linear classification in an infinite-dimensional feature space can be performed by means of operations in the lower-dimensional input space. Imagine all the power of infinite-dimensional hyperplanes, without the associated computational burden.

The purpose of this assignment is twofold: first, to work out, in detail, two simple classification problems in two-dimensional input space, one of them involving a mapping to a three-dimensional feature space; second, to provide some experience and some intuition on the capabilities of support vector machines.

2 Two simple examples

Consider the AND and XOR logic functions, defined in the following truth table:

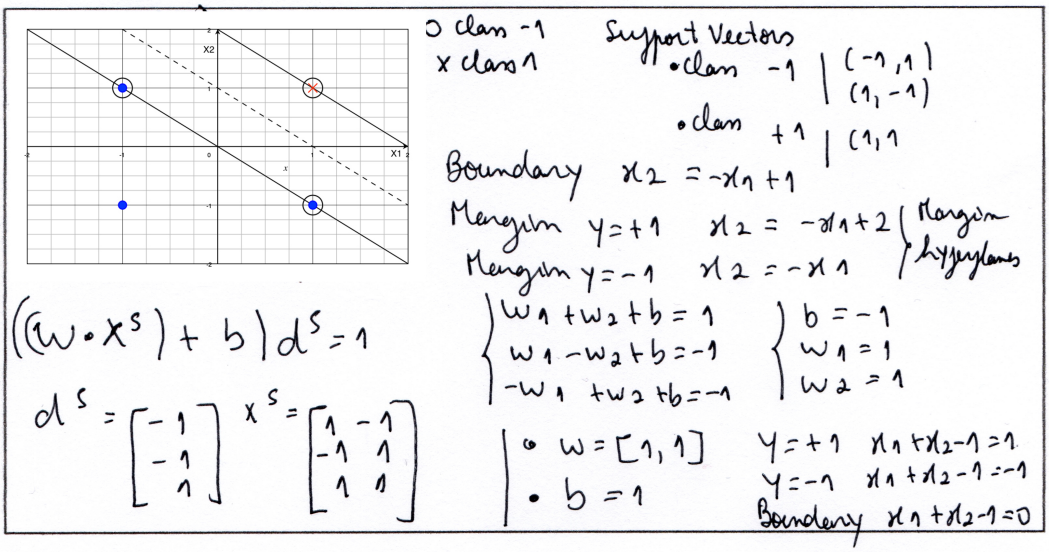
x_1	x_2	d_{AND}	d_{XOR}
-1	-1	-1	-1
-1	1	-1	1
1	-1	-1	1
1	1	1	-1

Here, the input pattern is a vector $\mathbf{x} = (x_1, x_2)$, and d_{AND} and d_{XOR} are the desired values for the AND and XOR functions. Note that, in this assignment, we represent logical *true* by 1 and logical *false* by -1. Similarly, in binary classification problems, we assign the desired value of 1 to the patterns of one of the classes, and the desired value of -1 to those of the other class.

2.1(T) For the AND function, find (by inspection) the maximum-margin separating straight line, the support vectors and the margin boundaries. Then compute the vector $\mathbf{w}$ and the bias b that satisfy the equation

$$(\mathbf{w} \cdot \mathbf{x}^s + b)\, d^s = 1 \tag{1}$$

for all support vectors $\mathbf{x}^s$, where d^s is the desired value corresponding to $\mathbf{x}^s$.[1]



2.2(T) Since, for the XOR function, a linear classification cannot be performed in the input space – explain why – we will consider here a simple nonlinear mapping to a three-dimensional feature space:

$$\tilde{\mathbf{x}} = \varphi(\mathbf{x}) = (x_1, x_2, x_1 x_2)^T . \tag{4}$$

Find the kernel function that corresponds to this mapping.

[1] It can be easily shown (but you're not asked to show) that, defining the border of the maximum-margin linear classifier by the equation

$$\widetilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} + b = 0, \tag{2}$$

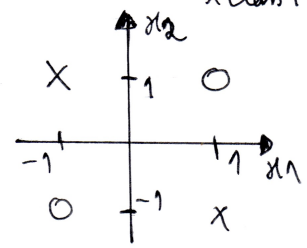
then $\widetilde{\mathbf{w}}$ and b obey the equation

$$(\widetilde{\mathbf{w}} \cdot \tilde{\mathbf{x}}^s + b)\, d^s = C \tag{3}$$

for all support vectors $\tilde{\mathbf{x}}^s$, where C is a constant. Normally, we choose $C = 1$.

In our case of the AND function, vectors with and without tilde are equal, since the feature space (where the linear classification is performed) is the input space itself.

R. XOR



$\tilde{x} = \phi(x) = (x_1, x_2, x_1 x_2)^T$

$= (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3)^T$

Kernel Function

$K(x, y) = \phi(x) \cdot \phi(y)$

$= (x_1, x_2, x_1 x_2) \cdot (y_1, y_2, y_1 y_2)$

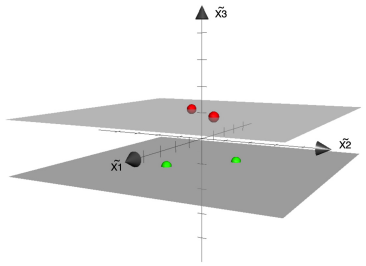
$= x_1 y_1 + x_2 y_2 + x_1 x_2 y_1 y_2$

The data is not separable by an hyperplane in the input space. We need to choose an appropriate transformation to separate by an hyperplane in the input space.

2.3(T) Visualize the points in this 3D feature space. Find, by inspection, which are the support vectors. Compute $\tilde{\mathbf{w}}$ and b in this feature space, so that equation (1) is satisfied for all support vectors.

R.

x_1	x_2	d xor	$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$
-1	-1	-1	-1	-1	1
-1	1	1	-1	1	-1
1	-1	1	1	-1	-1
1	1	-1	1	1	1



By inspection of the 3D Points in the 3D feature space, we conclude that all points are support vectors

class -1 $\{y = -1$ S.V $(-1,-1,1)(1,1,1)$

class +1 $\{y = +1$ S.V $(-1,1,-1)(1,-1,-1)$

$$\begin{cases} -\tilde{w}_1 - \tilde{w}_2 + \tilde{w}_3 + b = -1 \\ \tilde{w}_1 + \tilde{w}_2 + \tilde{w}_3 + b = -1 \\ -\tilde{w}_1 + \tilde{w}_2 - \tilde{w}_3 + b = 1 \\ \tilde{w}_1 - \tilde{w}_2 - \tilde{w}_3 + b = 1 \end{cases} \quad \begin{matrix} \tilde{w}_1 = 0 \\ \tilde{w}_2 = 0 \\ \tilde{w}_3 = -1 \\ b = 0 \end{matrix}$$

$\tilde{w} = [0 \; 0 \; -1] \quad b = 0$

$y = 1 \; \{-\tilde{x}_3 = 1 \Leftrightarrow \tilde{x}_3 = -1$

$y = -1 \; \{-\tilde{x}_3 = -1 \Leftrightarrow \tilde{x}_3 = 1$

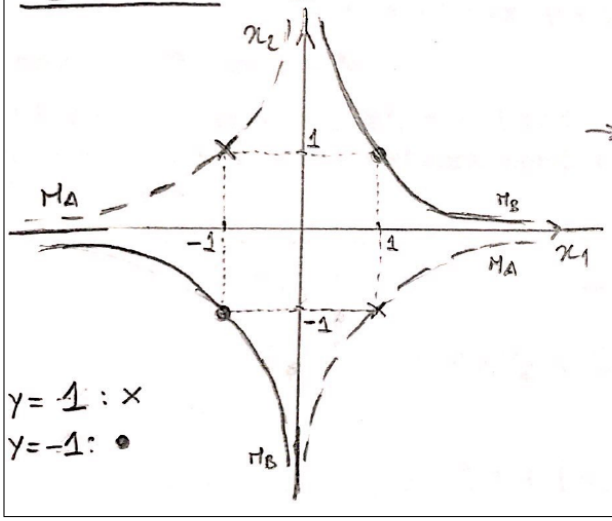
Boundary $-\tilde{x}_3 = 0 \Leftrightarrow \tilde{x}_3 = 0$

2.4(T) Algebraically express, in the two-dimensional input space, the classification border and the margin boundaries corresponding to the classifier found above for the XOR problem. Then sketch them in a graph, together with the input patterns.

R. Boundary: $\tilde{x}_3 = 0 \Rightarrow x_1 x_2 = 0 \Leftrightarrow \boxed{x_1 = 0 \vee x_2 = 0}$

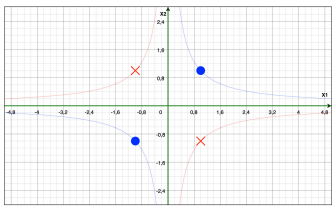
$M_A (y = +1)$: $\tilde{x}_3 = -1 \Rightarrow x_1 x_2 = -1 \Leftrightarrow \boxed{x_2 = -\frac{1}{x_1}}$

$M_B (y = -1)$: $\tilde{x}_3 = 1 \Rightarrow x_1 x_2 = 1 \Leftrightarrow \boxed{x_2 = \frac{1}{x_1}}$



→ Input Space

Boundary coincides with the axis



2.5(T) Indicate the mathematical condition under which the classifier that you have just developed will produce an output of 1. The condition should be expressed in terms of the input space coordinates. It shouldn't use coordinates from the feature space.

R. $Class = Sign[w\phi(x) + b]$

$= [Sign(-x_1 x_2)] \times c \;,\; c = 1$

$= -Sign(x_1 x_2)$ → For output of 1

$= \begin{cases} 1, & x_1 x_2 < 0 \\ 0, & x_1 x_2 = 0 \\ -1, & x_1 x_2 > 0 \end{cases}$ → $x_1 x_2 < 0 \Leftrightarrow (x_1 > 0 \wedge x_2 < 0) \vee (x_1 < 0 \wedge x_2 > 0)$

For $y = +1$: $(x_1 > 0 \wedge x_2 < 0) \vee (x_1 < 0 \wedge x_2 > 0)$

3 Classification using SVMs

A kernel commonly employed in pattern recognition problems is the polynomial one, defined by

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + a)^p - a^p, \tag{5}$$

where $a \in \mathbb{R}^+$ and $p \in \mathbb{N}$.[†]

3.1(T) Consider $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$, $a = 1$ and $p = 2$. Indicate the mapping to feature space that this kernel corresponds to, and the dimensionality of the feature space.

R. $K(x, Y) = (x \cdot Y + 1)^2 - 1$, for $a = 1$ and $p = 2$

$= [x_1 y_1 + x_2 y_2 + 1]^2 - 1$

$= x_1^2 y_1^2 + 2x_1 y_1 (x_2 y_2 + 1) + (x_2 y_2 + 1)^2 - 1$

$= x_1^2 y_1^2 + 2x_1 y_1 x_2 y_2 + 2x_1 y_1 + x_2^2 y_2^2 + 2x_2 y_2 + 1 - 1$

$= x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + 2x_1 y_1 + 2x_2 y_2 + x_2^2 y_2^2$

$= x_1^2 y_1^2 + \sqrt{2} x_1 x_2 (\sqrt{2} y_1 y_2) + \sqrt{2} x_1 (\sqrt{2} y_1) + \sqrt{2} x_2 (\sqrt{2} y_2) + x_2^2 y_2^2$

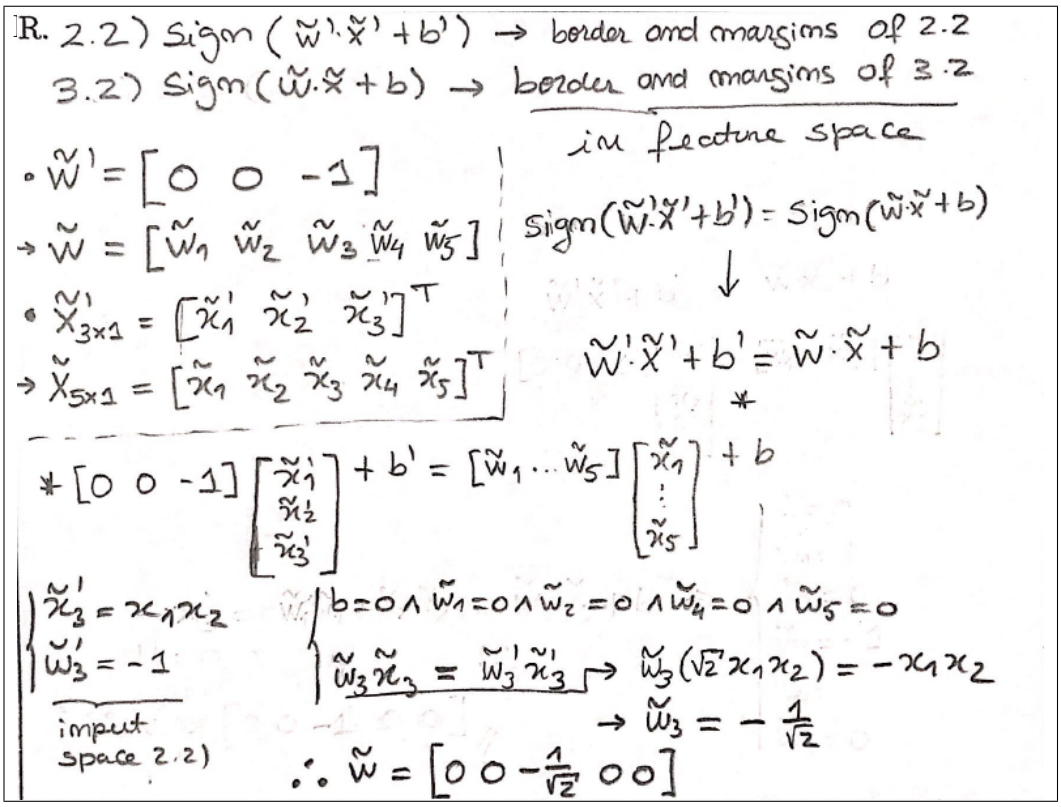
$= \phi(x) \cdot \phi(Y)$

$\Rightarrow \phi(x) = \begin{bmatrix} x_1^2 \\ \sqrt{2} x_1 \\ \sqrt{2} x_1 x_2 \\ \sqrt{2} x_2 \\ x_2^2 \end{bmatrix}$ $\phi(x): \mathbb{R}^2 \longrightarrow \mathbb{R}^5$

∴ The feature space has dimension = 5.

[†]This is one of the variants of the polynomial kernel. Another variant omits the term "$-a^p$" in the defining equation.

3.2(T) Assume again that $p = 2$ and $a = 1$. Find the vector $\widetilde{\mathbf{w}}$ that represents, in this new feature space, the same classification border and margins as in 2.2.



4 Experiments

The experimental part of this assignment uses the SVM toolbox from MatLab. Use function `svmtrain` for training the SVM and function `svmclassify` for testing (type help for more information on these functions). You will need to specify the 'kernel_function'. Use 'linear' for a linear classifier (*i.e.,* the feature space is equal to the input space), 'polynomial' for the kernel (5), where 'polyorder' stands for parameter p, and 'rbf' (radial basis function) for the kernel

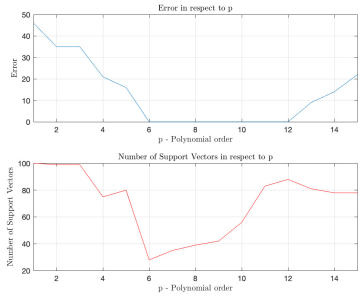
$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}}, \tag{6}$$

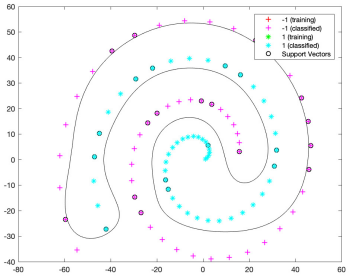
where 'rbf_sigma' stands for σ. Among these kernels, Gaussian RBF is the one that is most frequently used, for several reasons (for instance, because it is shift-invariant and isotropic).

Set the `svmtrain` parameter 'Method' to 'QP' to choose Quadratic Programming as the optimization method, and the 'boxconstraint' parameter to 10^4. This parameter corresponds to the soft margin penalty 'C' which specifies the relative weight of the margin violations in the objective function that is optimized in the training of the classifier.

When training and testing your classifiers, set option 'Showplot' to true in order to obtain the plot of the classification.

4.1(E) Load the file `spiral.mat`. This file contains the classical spiral example, with 50 patterns per class. Determine experimentally, using the polynomial kernel, the value of p for which you get the best classifier. (start with $p = 1$). Write down all experiments performed, together with the classification error percentages and number of support vectors (the support vectors can be obtained from the SVMStruct returned by svmtrain). Comment on the results you obtained.

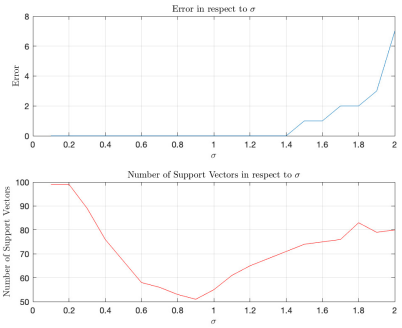


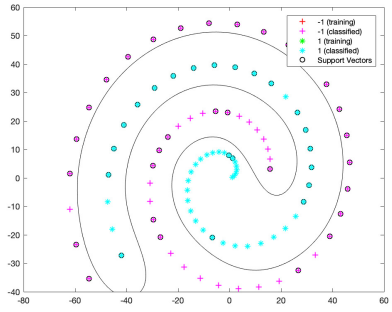


p	*Classification Error*	*No. of Support Vectors*
1	46%	100
2	35%	99
3	35%	99
4	21%	75
5	16%	80
6	**0%**	**28**
7	0%	35
8	0%	39
9	0%	42
10	0%	56
11	0%	83
12	0%	88
13	9%	81
14	14%	78
15	22%	78

The best classifier was obtained for p = 6, for which the error (0%) and number of support vectors (28) were minimal. Increasing too much the polynomial degree tends to cause overfitting in these problems. However a low degree (such as 1 or 2) doesn't work as well when the dataset has "bending" data, in this case, shaped as a spiral, as it is too underfitting. In short, p = 1 is a degree too low and p = 12 is too high for this problem, meaning that the best value should be in between, in this case it's p = 6, as proven by the graphs obtained.

4.2(E) Using the same data file (`spiral.mat`), try now the Gaussian RBF kernel. Find the approximate value of σ for which you can get the best classifier. Comment on the results you obtained.





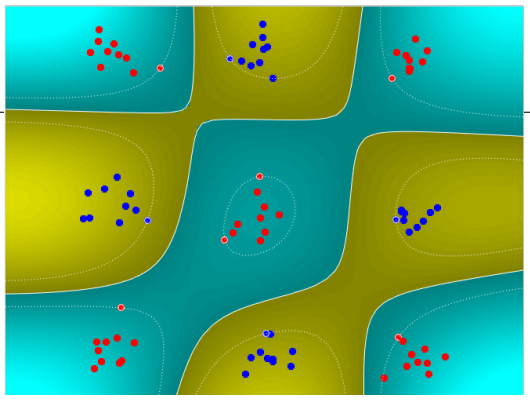
σ	*Classification Error*	*No. of Support Vectors*
0.1	0.0%	99
0.2	0.0%	99
0.3	0.0%	89
0.4	0.0%	76
0.5	0.0%	67
0.6	0.0%	58
0.7	0.0%	56
0.8	0.0%	53
0.9	**0.0%**	**51**
1.0	0.0%	55
1.1	0.0%	61
1.2	0.0%	65
1.3	0.0%	68
1.4	0.0%	71
1.5	1.0%	74
1.6	1.0%	75
1.7	2.0%	76
1.8	2.0%	83
1.9	3.0%	79
2	7.0%	80

The best classifier was obtained for σ = 0,9, for which the error (0%) and the number of support vectors (51) were minimal. For lower values of σ, the RBF has a smaller radius and, for that reason, the algorithm will try more to avoid misclassification, however that causes high overfitting and data points set more distant will be isolated, because the radius might be too small. For higher values of σ the algorithm will allow more misclassification, as the radius is bigger, which is not what's intended. Gaussian RBF has shown to be more flexible and better for data that can't be linearly separable, because it allows more possible parameters and classifiers for which we get proper classification, which is better for when we don't know exactly how the data is shaped beforehand. Moreover, the margins' size obtained for this best case are better, at the expense of choosing a kernel computationally more complex (higher number of support vectors).

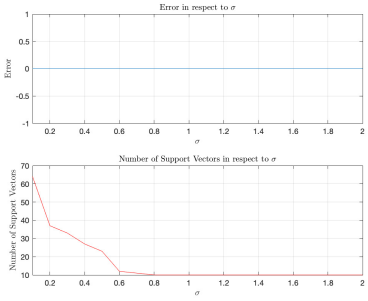
4.3(E) Load the file `chess33.mat` and set 'boxconstraint' parameter to Inf to enforce a hard margin SVM, for separable data. Using the Gaussian RBF kernel, find a value of σ that approximately minimizes the number of support vectors, while correctly classifying all patterns. Indicate the value of σ and the number of support vectors.

R. *σ = 0,9 | Error = 0,0% | N. Support Vectors = 10*

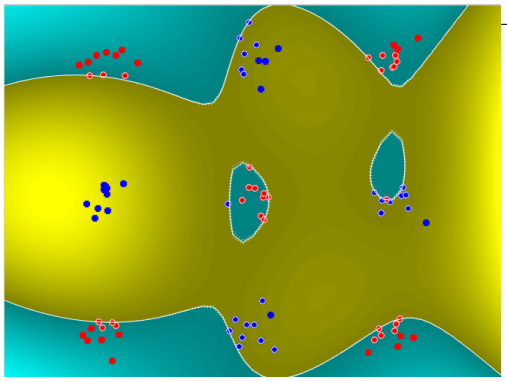
The number of support vectors tends to 10, we chose σ = 1 as it outputs the minimum number of support vectors while the error is still 0%.



This result was obtained with a different MATLAB version from the one we used to select the best value of sigma, however the behaviour is very similar with both versions.



4.4(E) Load the file `chess33n.mat` which is similar to the one used in the previous question, except for the presence of a couple of outlier patterns. Run the classification algorithm on these data with the same value of σ, and comment on how the results changed, including the shape of the classification border, the margin size and the number of support vectors.



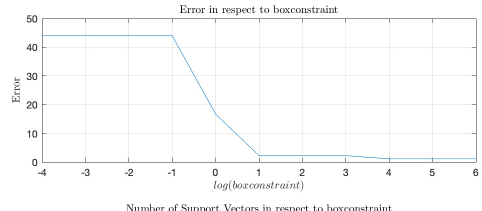
σ = 0,9 | Error = 0,0% | N. Support Vectors = 54

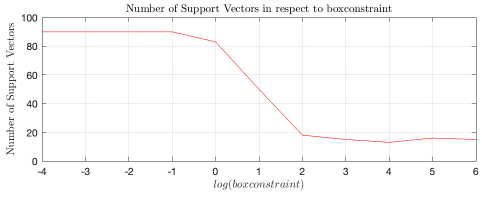
With the presence of outliers, for the same value σ, the error remained 0% and the number of support vectors increased from 10 to 54. The margins size was considerably reduced, at least 2400 times, due to the outlier present very close to class 1 (middle cluster).

The obtained border is more bended so it can allow the slack and influence of outliers. The number of support vectors also increased as a consequence of the trade-off of getting a more complex model, with some overfitting, at the expense of a smaller margin, in order to have less classification error (0%) and due to the more bended border.

4.5(E) Now reduce the value of 'boxconstraint' parameter in order to obtain the so-called *soft margin* SVM. Try different values (suggestion: use powers of 10) and comment on the results.

For σ = 0,9, reducing the value of the boxconstraint, the values of error and number of support vectors are maximum for small powers of 10. For large values of boxconstraint (C) the optimization problem chooses a smaller margin hyperplane, if the hyperplane classifies better the training data. Contrarily, smaller values of boxconstraint allow more misclassification of the training data, since the optimization problem chooses a hyperplane with larger margins, this explains why the error is greater for smaller powers of 10. Due to the presence of outliers a reasonable trade-off between allowing more misclassification (larger margins) or being more strict and fitting with smaller margins and less misclassification is needed. In short, the higher the boxconstraint, the higher is the cost of the misclassified points, leading to a more strict separation of the data. As a result, a soft margin SVM tends to misclassify more problems with outliers (higher error).





$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \quad s.t. \quad y^{(i)}(x^{(i)}\cdot w + b) - 1 + \xi_i \geq 0,\ \forall i.$$

Box Constraint	*Classification Error*	*No. of Support Vectors*
10^{-4}	44%	90
10^{-3}	44%	90
10^{-2}	44%	90
10^{-1}	44%	90
1	16.7%	83
10	2.2%	50
10^2	2.2%	18
10^3	2.2%	15
10^4	1.1%	13
10^5	1.1%	16
10^6	1.1%	15

This table was obtained using the SVM tool.