Instituto Superior Técnico

Departamento de Engenharia Electrotécnica e de Computadores

Machine Learning

5th Lab Assignment

Shift Thursday 17h Group number 5

Number 80966 Name Franciisco Azevedo

Number 81356 Name Duarte Dias

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:

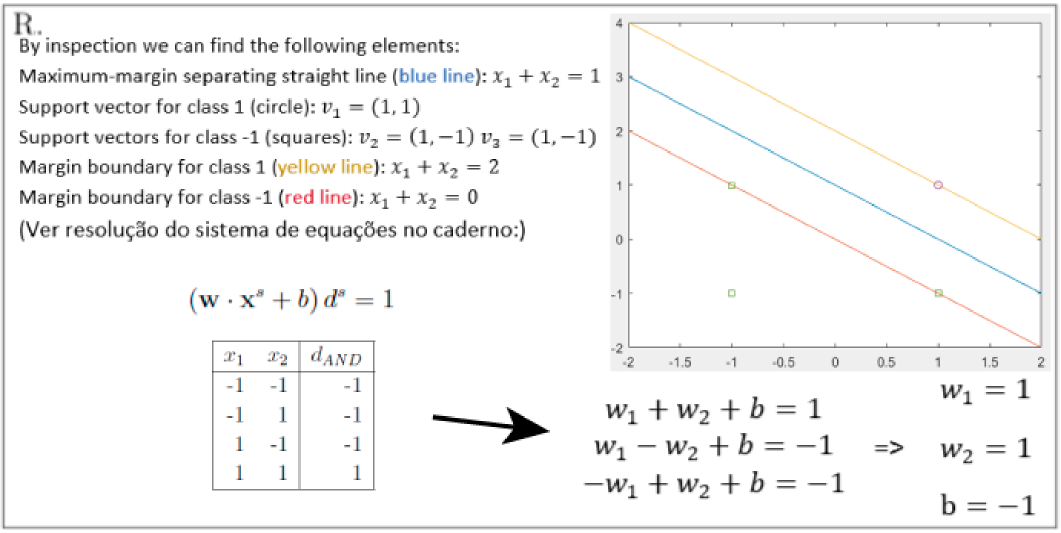
x_1	x_2	d_{AND}	d_{XOR}
-1	-1	-1	-1
-1	1	-1	1
1	-1	-1	1
1	1	1	-1

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]



R.

By inspection we can find the following elements:

Maximum-margin separating straight line (blue line): $x_1 + x_2 = 1$

Support vector for class 1 (circle): $v_1 = (1, 1)$

Support vectors for class -1 (squares): $v_2 = (1, -1)$ $v_3 = (1, -1)$

Margin boundary for class 1 (yellow line): $x_1 + x_2 = 2$

Margin boundary for class -1 (red line): $x_1 + x_2 = 0$

(Ver resolução do sistema de equações no caderno:)

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

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

$$\longrightarrow \quad \begin{array}{r} w_1 + w_2 + b = 1 \\ w_1 - w_2 + b = -1 \\ -w_1 + w_2 + b = -1 \end{array} \quad \Rightarrow \quad \begin{array}{l} w_1 = 1 \\ w_2 = 1 \\ \text{b} = -1 \end{array}$$

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

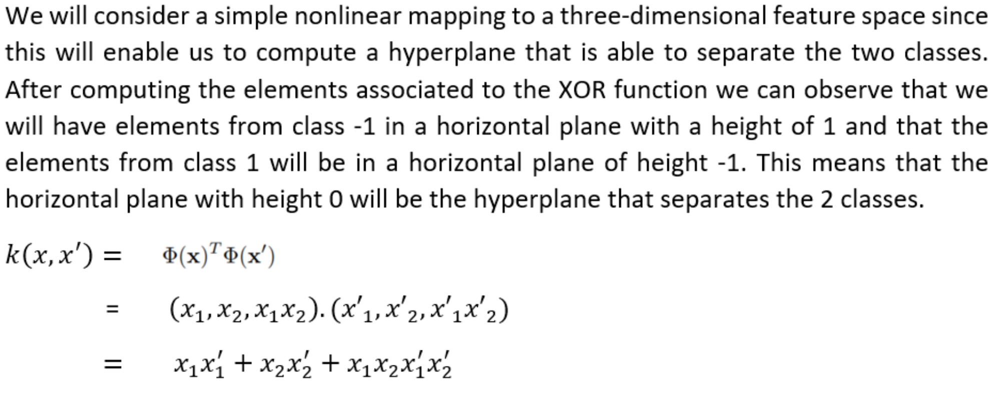
$$\widetilde{\mathbf{w}} \cdot \tilde{\mathbf{x}} + b = 0, \tag{2}$$

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 $C = 1$.

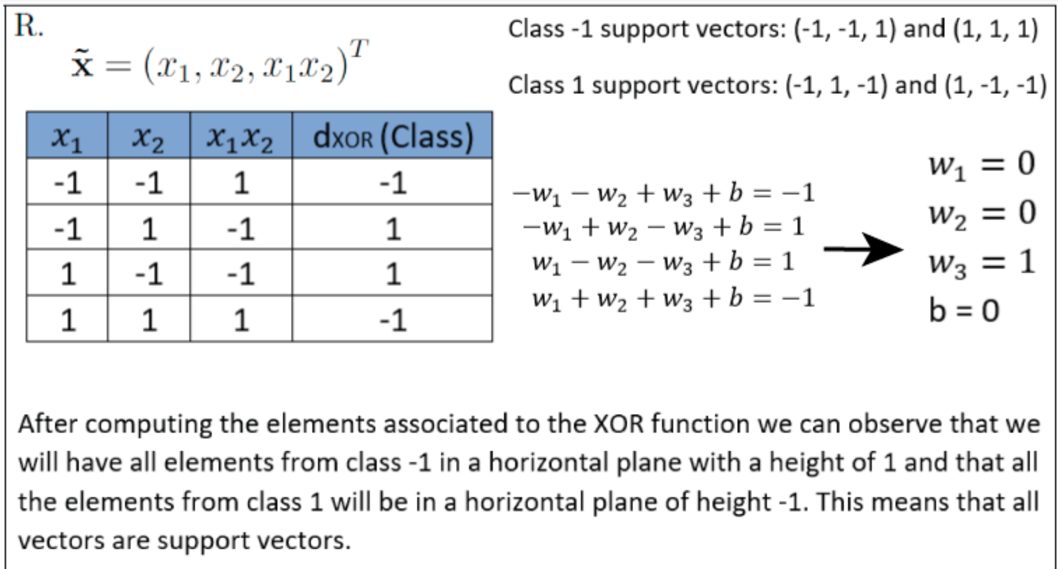
In our case of the AND function, vectors with and without tilde are equal, since the feature space (where the linear classification is performed) is the input space itself.

R. 

We will consider a simple nonlinear mapping to a three-dimensional feature space since this will enable us to compute a hyperplane that is able to separate the two classes. After computing the elements associated to the XOR function we can observe that we will have elements from class -1 in a horizontal plane with a height of 1 and that the elements from class 1 will be in a horizontal plane of height -1. This means that the horizontal plane with height 0 will be the hyperplane that separates the 2 classes.

$$\begin{aligned} k(x,x') &= \Phi(\mathbf{x})^T\Phi(\mathbf{x}') \\ &= (x_1, x_2, x_1x_2).(x'_1, x'_2, x'_1x'_2) \\ &= x_1x'_1 + x_2x'_2 + x_1x_2x'_1x'_2 \end{aligned}$$

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



R.

$$\tilde{\mathbf{x}} = (x_1, x_2, x_1x_2)^T$$

Class -1 support vectors: (-1, -1, 1) and (1, 1, 1)

Class 1 support vectors: (-1, 1, -1) and (1, -1, -1)

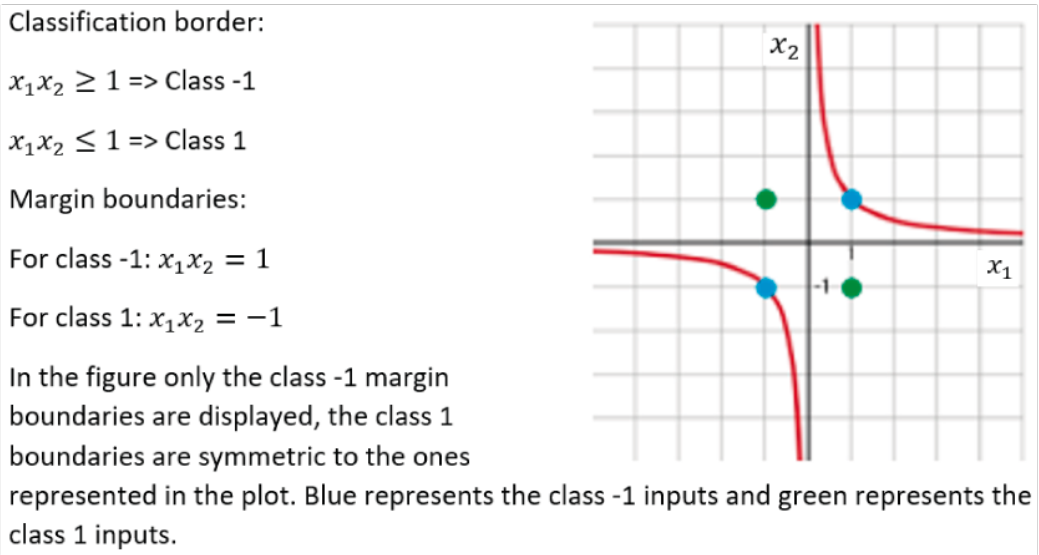
x_1	x_2	x_1x_2	dXOR (Class)
-1	-1	1	-1
-1	1	-1	1
1	-1	-1	1
1	1	1	-1

$$\begin{aligned} -w_1 - w_2 + w_3 + b &= -1 \\ -w_1 + w_2 - w_3 + b &= 1 \\ w_1 - w_2 - w_3 + b &= 1 \\ w_1 + w_2 + w_3 + b &= -1 \end{aligned} \longrightarrow \begin{aligned} w_1 &= 0 \\ w_2 &= 0 \\ w_3 &= 1 \\ b &= 0 \end{aligned}$$

After computing the elements associated to the XOR function we can observe that we will have all elements from class -1 in a horizontal plane with a height of 1 and that all the elements from class 1 will be in a horizontal plane of height -1. This means that all vectors are support vectors.

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.



Classification border:

$x_1x_2 \geq 1$ => Class -1

$x_1x_2 \leq 1$ => Class 1

Margin boundaries:

For class -1: $x_1x_2 = 1$

For class 1: $x_1x_2 = -1$

In the figure only the class -1 margin boundaries are displayed, the class 1 boundaries are symmetric to the ones represented in the plot. Blue represents the class -1 inputs and green represents the class 1 inputs.

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.

$x_1x_2 \geq 1$

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.

This kernel corresponds to a mapping from a 2-D space into a 3-D space. The actual mapping to feature space is implicit but can be derived. If we assume that the kernel is the basic formulation (ignoring the extra term) it is defined by:

$$K(x, y) = (x^T y + a)^p$$

The kernel K corresponds to an inner product in a feature space based on some mapping ϕ:

$$K(x, y) = \langle \varphi(\mathrm{x}), \varphi(\mathrm{y}) \rangle$$

For p = 2 we have the quadratic kernel and, after expanding the equation we get:

$$K(x,y) = \left(\sum_{i=1}^{n} x_i y_i + a \right)^2 = \left(\sum_{i=1}^{n} x_i^2 y_i^2 \right) + \sum_{i=1}^{n} \sum_{j=1}^{i-1} (\sqrt{2} x_i x_j)(\sqrt{2} y_i y_j) + \sum_{i=1}^{n} (y_i \sqrt{2a})(x_i \sqrt{2a}) + a^2$$

So from this it follows that the feature map is:

$$\varphi(\mathrm{x}) = \langle x_n^2, \ldots, x_1^2; \sqrt{2x_n} x_{n-1}, \ldots, \sqrt{2x_n} x_1, \sqrt{2x_{n-1}} x_{n-2}, \ldots, \sqrt{2x_{n-1}} x_1, \ldots, \sqrt{2x_2} x_1;\ \sqrt{2a} x_n, \ldots, \sqrt{2a} x_1; a \rangle$$

[†] 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.

R.

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

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

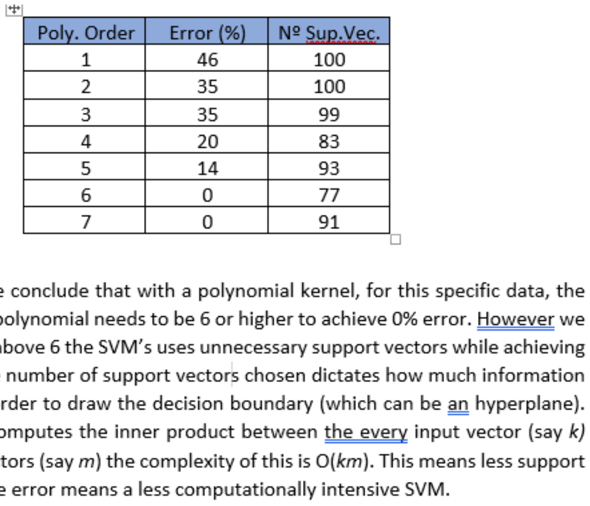
where ’rbf_sigma’ stands for σ. Among these kernels, Gaussian RBF is the one that is most frequently used, for several reasons (for instance, because it is shift-invariant and isotropic).

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.

R.

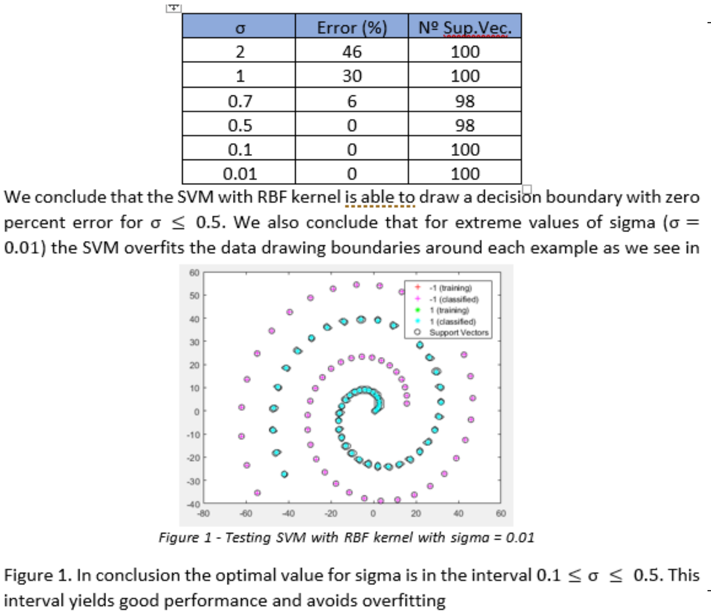


Poly. Order	Error (%)	Nº Sup.Vec.
1	46	100
2	35	100
3	35	99
4	20	83
5	14	93
6	0	77
7	0	91

From the results we conclude that with a polynomial kernel, for this specific data, the order needs of the polynomial needs to be 6 or higher to achieve 0% error. However we see that for orders above 6 the SVM's uses unnecessary support vectors while achieving the same error. The number of support vectors chosen dictates how much information the SVM needs in order to draw the decision boundary (which can be an hyperplane). Given that SVM's computes the inner product between the every input vector (say *k)* and the support vectors (say *m*) the complexity of this is O(*km*). This means less support vectors for the same error means a less computationally intensive SVM.

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.

R.



σ	Error (%)	Nº Sup.Vec.
2	46	100
1	30	100
0.7	6	98
0.5	0	98
0.1	0	100
0.01	0	100

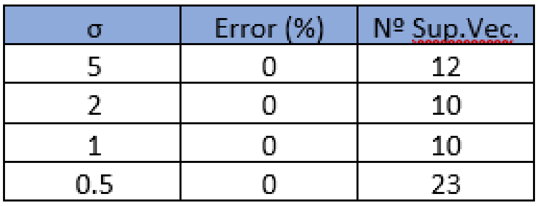
We conclude that the SVM with RBF kernel is able to draw a decision boundary with zero percent error for σ ≤ 0.5. We also conclude that for extreme values of sigma (σ = 0.01) the SVM overfits the data drawing boundaries around each example as we see in

Figure 1 - Testing SVM with RBF kernel with sigma = 0.01

Figure 1. In conclusion the optimal value for sigma is in the interval 0.1 ≤ σ ≤ 0.5. This interval yields good performance and avoids overfitting

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.

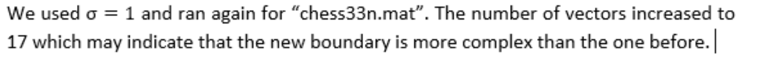
R.



σ	Error (%)	Nº Sup.Vec.
5	0	12
2	0	10
1	0	10
0.5	0	23

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.

R.



We used σ = 1 and ran again for “chess33n.mat”. The number of vectors increased to 17 which may indicate that the new boundary is more complex than the one before.|

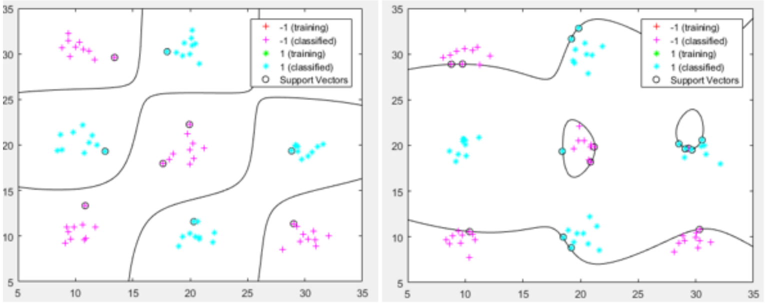
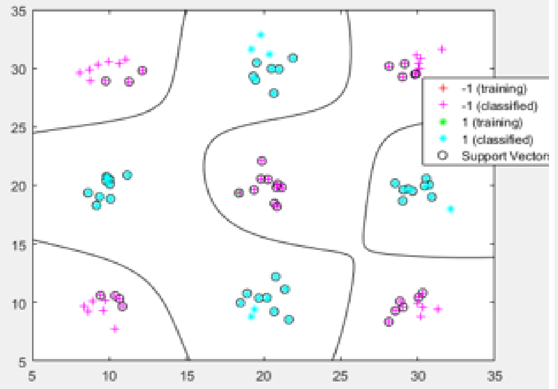


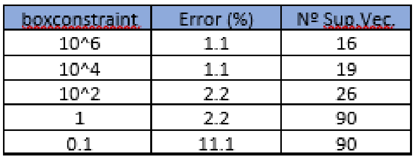
Figure 2 - Decision boundary of SVM with RBF Kernel and sigma = 1. On the left is "chess33.mat" and on the right "chess33n.mat"

Figure 2 shows that the shape of the boundary changed drastically and the margin aswell. This means that the SVM with RBF Kernel and sigma = 1 is very sensitive to small variation in its input data. Small differences in the input data yield drastic differences in the decision boundary. In this case, just changing some points to outliers creates a decision boundary with minimal to almost no margin that clearly overfits the data.

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.

R. In this exercise we reduced the boxconstraint value to achieve a soft margin. A soft margin is when we prefer a decision boundary that separates the bulk of the data while ignoring a few outliers (that otherwise would increase the classification error). We show below the results for several boxconstraint values. For values lower than 10^7 the error percentage is not zero so its already adapting to a soft margin. As we decrease the boxconstraint value we obtain larger margins and a decision boundary that doesn’t overfit the data. This means that the soft margin ignores the outliers and provides a solution that generalizes well to new data. The figure below shows a decision boundary for boxconstraint = 10^1 We can see that using boxconstraint = 10^1 the decision boundary approximates the one obtained when not using boxconstraint.





boxconstraint	Error (%)	Nº Sup.Vec.
10^6	1.1	16
10^4	1.1	19
10^2	2.2	26
1	2.2	90
0.1	11.1	90

The optimal value for the trade-off of performance and overfit is sigma = 10^2.