Instituto Superior Técnico

Departamento de Engenharia Electrotécnica e de Computadores

Machine Learning

5^{th} Lab Assignment

Shift Wed - 14h Group number 1

Number 78486 Name Luís Bernardo de Brito Mendes Rei

Number 78761 Name João Miguel Limpo de Lacerda Pestana Girão

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.

Upon inspecting the points in the input space (figure 1), it's clear that an hyperplane can be used to divide both classes using as support vectors the points (1,1) from class 1 and (1,-1), (-1,1) from class -1.

Replacing in (1) x^s and d^s by the points and classes above, we get the following equation system:

$$\begin{cases} \omega_1 + \omega_2 + b = 1 \\ \omega_1 + -\omega_2 + b = -1 \\ -\omega_1 + \omega_2 + b = -1 \end{cases} \Leftrightarrow \begin{cases} \omega_1 = \omega_2 - b - 1 \\ \omega_2 = \omega_1 - b - 1 \\ b = 1 - \omega_1 - \omega_2 \end{cases} \Leftrightarrow \begin{cases} \omega_1 = 1 \\ \omega_2 = 1 \\ b = -1 \end{cases}$$

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.

Contrary to what happened with the previous example, the points in the input space for the XOR function (figure 2) can't be separated as there is no line equation that divides the plane in two parts where each one only holds one class.

A higher dimension feature space is then considered so that an hyperplane can be used to split both classes. This feature space is defined by $\tilde{x} = \varphi(x) = (x_1, x_2, x_1x_2)^T$, and, as all feature spaces, its kernel function of x and y is given by the inner product of $\tilde{x}$ and $\tilde{y}$, i.e

$$K(x, y) = \varphi(x) \cdot \varphi(y) = x_1y_1 + x_2y_2 + x_1x_2y_1y_2.$$

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.

With (4), the points in the input space are all translated along the new axis by 1 or -1 according to their class (figure 3). We can then analyze the points in feature space (table 1) and select the ones closest to the points in the other class (table 2) as support vectors.

Using these points in (1) again we get

$$\begin{cases} -\omega_1 - \omega_2 + \omega_3 + b = -1 \\ -\omega_1 + \omega_2 - \omega_3 + b = 1 \\ \omega_1 - \omega_2 - \omega_3 + b = 1 \\ \omega_1 + \omega_2 + \omega_3 - b = -1 \end{cases} \Leftrightarrow \begin{cases} \omega_1 = -\omega_2 + \omega_3 + b + 1 \\ \omega_2 = \omega_1 + \omega_3 - b + 1 \\ \omega_3 = \omega_1 - \omega_2 + b - 1 \\ b = -\omega_1 - \omega_2 - \omega_3 - 1 \end{cases} \Leftrightarrow \begin{cases} \omega_1 = 0 \\ \omega_2 = 0 \\ \omega_3 = -1 \\ b = 0 \end{cases}$$

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.

Now that we have the mapping function $\varphi(x)$ and the parameters $\tilde{\omega}$ and b we can compute the equations for the classification border and margin boundaries (figure 2a). For the classification border we have $\tilde{\omega} \cdot \tilde{x} + b = 0$ (a) while for the margin boundaries the equation is $(\tilde{\omega} \cdot \tilde{x}^s + b)d^s = 1$ (b), with $\tilde{\omega} = (0, 0, -1)^T$, $\tilde{x}^s = (x_1^s, x_2^s, x_1^s x_2^s)^T$, $b = 0$ and $d^s = 1 \vee d^s = -1$.

From (a) and (b) we have:

$$\begin{cases} x_1 x_2 = 0 & \text{, for the classification border} \\ x_1 x_2 = -1 & \text{, for the positive margin boundary} \\ x_1 x_2 = 1 & \text{, for the negative margin boundary} \end{cases} \Leftrightarrow \begin{cases} x_1 = 0 \vee x_2 = 0 & \text{, for the classification border} \\ x_2 = -\frac{1}{x_1} & \text{, for the positive margin boundary} \\ x_2 = \frac{1}{x_1} & \text{, for the negative margin boundary} \end{cases}$$

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.

After determining the equations for the margin boundaries, one can generalize them for classification of points outside the training set. For this, we will take equation (3) and modify it so that we have a general point instead of a support vector,

$$(\tilde{\omega} \cdot \tilde{x} + b)d = C. \tag{3a}$$

We can use this modified equation to compute the class of the point to be classified by looking at the sign of d (positive for class 1 and negative for class -1). Using (3a) and isolating the variable d:

$$(-x_1 x_2)d = C \Leftrightarrow d = -\frac{C}{x_1 x_2} \Leftrightarrow d = sign(-\frac{C}{x_1 x_2}) *$$

* Since we want to obtain $d = 1$ we'll need $sign(x_1) \neq sign(x_2)$, which corresponds to the 1_{st} and 3_{rd} quadrants.

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.

Assuming $a = 1$, $p = 2$, $x = (x_1, x_2)^T$ and $y = (y_1, y_2)^T$ then the kernel given in (5) will have the form:

$$K(x, y) = (x \cdot y + a)^p - a^p =$$

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

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

And we know that $K(x, y)$ is given by the inner product of $\tilde{x}$ by $\tilde{y}$, so we have:

$$K(x, y) = \varphi(x) \cdot \varphi(y) = \begin{bmatrix} \sqrt{2}x_1 & \sqrt{2}x_2 & \sqrt{2}x_1 x_2 & x_1^2 & x_2^2 \end{bmatrix} \begin{bmatrix} \sqrt{2}y_1 \\ \sqrt{2}y_2 \\ \sqrt{2}y_1 y_2 \\ y_1^2 \\ y_2^2 \end{bmatrix}.$$

We can then conclude that $\varphi(x)$ is a mapping from an input space with 2 dimensions to a feature space with 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.

R. From 2.2 we know the equations for the margin boundaries and the classification border, as well as the support vectors for those boundaries. As stated in the previous question, the feature space has dimension 5, meaning there are 6 unknowns ($\tilde{\omega}$ and b) for which we already have 4 equations, derived from the support vectors and margin formulas. For simplification, let us choose two simple points on the classification border $x^{b1} = (1, 0)$ and $x^{b2} = (0, 1)$ and add them to our list of points ($x^{s1} = (-1, -1)$, $x^{s2} = (-1, 1)$, $x^{s3} = (1, -1)$, $x^{s4} = (1, 1)$). Then, using equations (1) and (2) and the transformation function $\tilde{x} = \varphi(x) = \begin{bmatrix}\sqrt{2}x_1 & \sqrt{2}x_2 & \sqrt{2}x_1x_2 & x_1^2 & x_2^2\end{bmatrix}$ on these 6 points we get the following equation system:

$$f(x) = \begin{cases} \sqrt{2}x_1^{s1}\omega_1 + \sqrt{2}x_2^{s1}\omega_2 + \sqrt{2}x_1^{s1}x_2^{s1}\omega_3 + (x_1^{s1})^2\omega_4 + (x_2^{s1})^2\omega_5 + b = -1 & (i) \\ \sqrt{2}x_1^{s2}\omega_1 + \sqrt{2}x_2^{s2}\omega_2 + \sqrt{2}x_1^{s2}x_2^{s2}\omega_3 + (x_1^{s2})^2\omega_4 + (x_2^{s2})^2\omega_5 + b = 1 & (ii) \\ \sqrt{2}x_1^{s3}\omega_1 + \sqrt{2}x_2^{s3}\omega_2 + \sqrt{2}x_1^{s3}x_2^{s3}\omega_3 + (x_1^{s3})^2\omega_4 + (x_2^{s3})^2\omega_5 + b = 1 & (iii) \\ \sqrt{2}x_1^{s4}\omega_1 + \sqrt{2}x_2^{s4}\omega_2 + \sqrt{2}x_1^{s4}x_2^{s4}\omega_3 + (x_1^{s4})^2\omega_4 + (x_2^{s4})^2\omega_5 + b = -1 & (iv) \\ \sqrt{2}x_1^{b1}\omega_1 + \sqrt{2}x_2^{b1}\omega_2 + \sqrt{2}x_1^{b1}x_2^{b1}\omega_3 + (x_1^{b1})^2\omega_4 + (x_2^{b1})^2\omega_5 + b = 0 & (v) \\ \sqrt{2}x_1^{b2}\omega_1 + \sqrt{2}x_2^{b2}\omega_2 + \sqrt{2}x_1^{b2}x_2^{b2}\omega_3 + (x_1^{b2})^2\omega_4 + (x_2^{b2})^2\omega_5 + b = 0 & (vi) \end{cases}$$

Applying the results obtained in 2.4, namely $x_2 = -\frac{1}{x_1}$ on (ii) and (iii), $x_2 = \frac{1}{x_1}$ on (i) and (iv), $x_1 = 0$ on (v) and $x_2 = 0$ on (vi), $f(x)$ takes the form

$$f(x) = \begin{cases} \sqrt{2}x_1^{s1}\omega_1 + \frac{\sqrt{2}}{x_1^{s1}}\omega_2 + \sqrt{2}\frac{x_1^{s1}}{x_1^{s1}}\omega_3 + (x_1^{s1})^2\omega_4 + \frac{1}{(x_1^{s1})^2}\omega_5 + b = -1 & (i) \\ \sqrt{2}x_1^{s2}\omega_1 - \frac{\sqrt{2}}{x_1^{s2}}\omega_2 - \sqrt{2}\frac{x_1^{s2}}{x_1^{s2}}\omega_3 + (x_1^{s2})^2\omega_4 + \frac{1}{(x_1^{s2})^2}\omega_5 + b = 1 & (ii) \\ \sqrt{2}x_1^{s3}\omega_1 - \frac{\sqrt{2}}{x_1^{s3}}\omega_2 - \sqrt{2}\frac{x_1^{s3}}{x_1^{s3}}\omega_3 + (x_1^{s3})^2\omega_4 + \frac{1}{(x_1^{s3})^2}\omega_5 + b = 1 & (iii) \\ \sqrt{2}x_1^{s4}\omega_1 + \frac{\sqrt{2}}{x_1^{s4}}\omega_2 + \sqrt{2}\frac{x_1^{s4}}{x_1^{s4}}\omega_3 + (x_1^{s4})^2\omega_4 + \frac{1}{(x_1^{s4})^2}\omega_5 + b = -1 & (iv) \\ \sqrt{2}x_2^{b1}\omega_2 + (x_2^{b1})^2\omega_5 + b = 0 & (v) \\ \sqrt{2}x_1^{b2}\omega_1 + (x_1^{b2})^2\omega_4 + b = 0 & (vi) \end{cases}$$

Replacing x by the values on the list and solving for $\tilde{\omega}$ and b we obtain $\tilde{\omega} = \begin{bmatrix}0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0\end{bmatrix}$ and $b = 0$.

Note that replacing $\tilde{x}$, $\tilde{\omega}$ and b in (3) we get the equations for the border/margins obtained 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.

R.

After training the classifiers with power p from 1 to 10 and testing them using the same training points we got the results shown on table 3 in annex. Observing them we take that a good classifier for this example should have a power never lower than 5, since it will misclassify some training points∗, as the power isn't big enough to obtain a good estimate of the different class regions. The error percentage is the same for all values of p bigger than 5, but the number of support vectors is significantly smaller for $p = 6$ with only 35 support vectors (less than half of the second smallest value). From this we conclude that the best classifier is the one with power value 6, this way there is no overfit while maintaining a good estimate of the class regions (figure 4).

∗ Misclassification in the training set isn't necessarily a bad thing, but given the distribution of our class points and the fact that we're using a hard margin SVM, it is highly recommended that we have no errors if we want to obtain a good classifier.

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.

Using the Gaussian RBF kernel with small enough σ one gets no errors, because the regions contain the point that originated them in their middle*, but, as the value of the variance grows the regions start overlapping and with big enough values it is possible to notice that the regions converge to the areas with bigger concentration of class points (figure 8) and eventually converges to what resembles a separation using an hyperplane in $\mathcal{R}^2$ (figure 9) . Since there is no error in the tested points the method for evaluating the classifiers must be different, or at least have more criteria, and so, with that in mind, we added one criterion: the more margin of error for both classes there is, the better the classifier. Of the 10 values of σ we tested, 3 of them stood out after we added the previous criterion $\sigma = 0.1$, $\sigma = 0.15$ and $\sigma = 0.2$ (figures 5 to 7). Within this trio we opted to choose $\sigma = 0.15$ as it has a good error margin both on the beginning of the spiral as on its outer part.

$*$This can be considered overfitting, as the variance is so small the classifier considers that the points from class $+1$ are separated into diferent regions, when in reality, it's all the same region.

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.

The values used for training and testing of the classifier are presented in Table 5 in annex. After sweeping through those values we found the approximate optimal value of σ to be 1 (figure 10), which results in 10 support vectors. It has less support vectors and is more similar to the desired pattern than other values in its vicinity, with the region distribution faintly resembling a checkered pattern when $\sigma = 0.25$ (figure 11) and starting to look less like a chess board for values bigger than 2.5 (figure 12). On a side note, a hard margin SVM can be useful in cases where there are no outliers but it usually doesn't fair well in their presence.

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.

A hard margin SVM has its support vectors on the classification border, which affects immensely the class regions whenever outliers appear, since it doesn't allow any errors in the training set and may lead to overfitting. This can be observed in figure 13 in annex, where it can be seen how these outliers changed the best guess for the classification regions obtained in the previous question. We can see that the addition of a few outliers distorted the classification border and margins so much that the region distribution does not resemble in the least a checkered pattern. It can also be observed that the margins are much smaller than when there were no outliers, as the support vectors are extremely close to the classification border, and as such there is a need for more support vectors to find the equations for the margin boundaries.

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.

When using a soft margin SVM, we allow some misclassifications to occur when computing the classification border and margin boundaries, making the algorithm more robust to outliers. A higher value for the 'boxconstraint' variable when calling the *svmtrain* function is equivalent to "hardening" the margin of the support vector machine. The optimal solution found (figure 14) is not as good as when there were no outliers. In figures 15 and 16 it can be seen that the choice of hardness affects greatly the outcome of the training: a low enough value allows too many errors while a high enough value introduces some overfitting.

ANNEX

Table 1 – Classification of the XOR function in the feature space.

x_1	x_2	x_1x_2	d_{XOR}
-1	-1	1	-1
-1	1	-1	1
1	-1	-1	1
1	1	1	-1

Table 2 - Distance to the closest point of the different class.

x_1	x_2	x_1x_2	n_{xy}
-1	-1	1	$2\sqrt{2}$
-1	1	-1	$2\sqrt{2}$
1	-1	-1	$2\sqrt{2}$
1	1	1	$2\sqrt{2}$

Table 3 – Polynomial classifier error and support vectors.

p	error(%)	sv
1	46	100
2	35	100
3	35	99
4	21	78
5	16	91
6	0	35
7	0	82
8	0	92
9	0	96
10	0	95

Table 4 – Gaussian RBF support vectors.

σ	sv
0.010	100
0.025	100
0.050	99
0.075	99
0.100	100
0.150	99
0.200	99
0.250	100
0.500	82
1.000	56

Table 5 – Gaussian RBF support vectors.

σ	SV
0.0001	90
0.0025	90
0.0050	90
0.0075	90
0.0100	90
0.0250	90
0.0500	87
0.0750	82
0.1000	64
0.2500	31
0.5000	23
0.7500	16
1.0000	10
2.5000	12
5.0000	12
6.0000	18

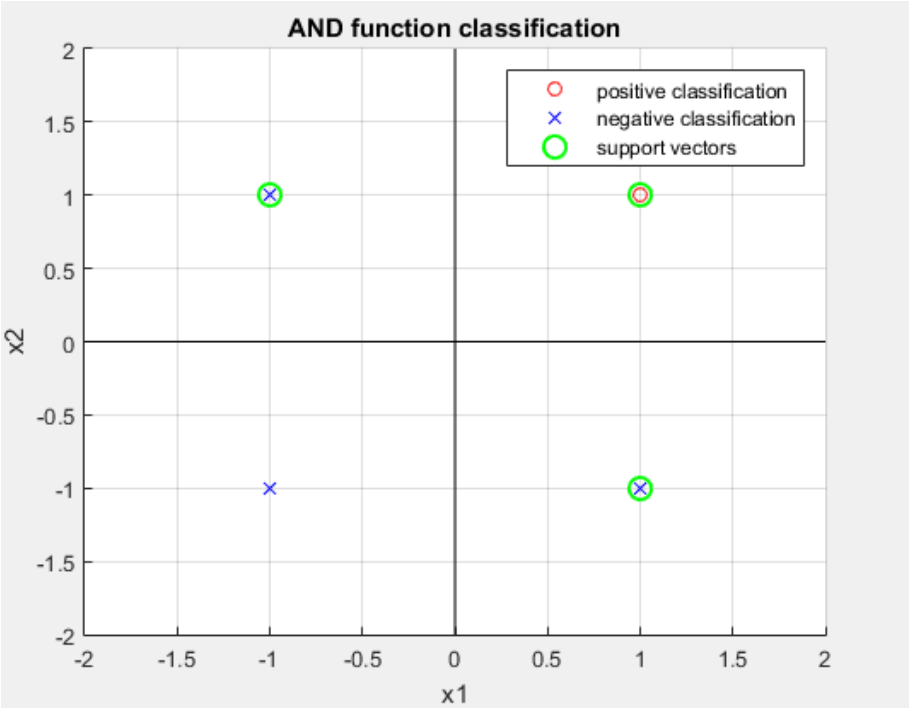

Figure 1 – AND classification in input space.

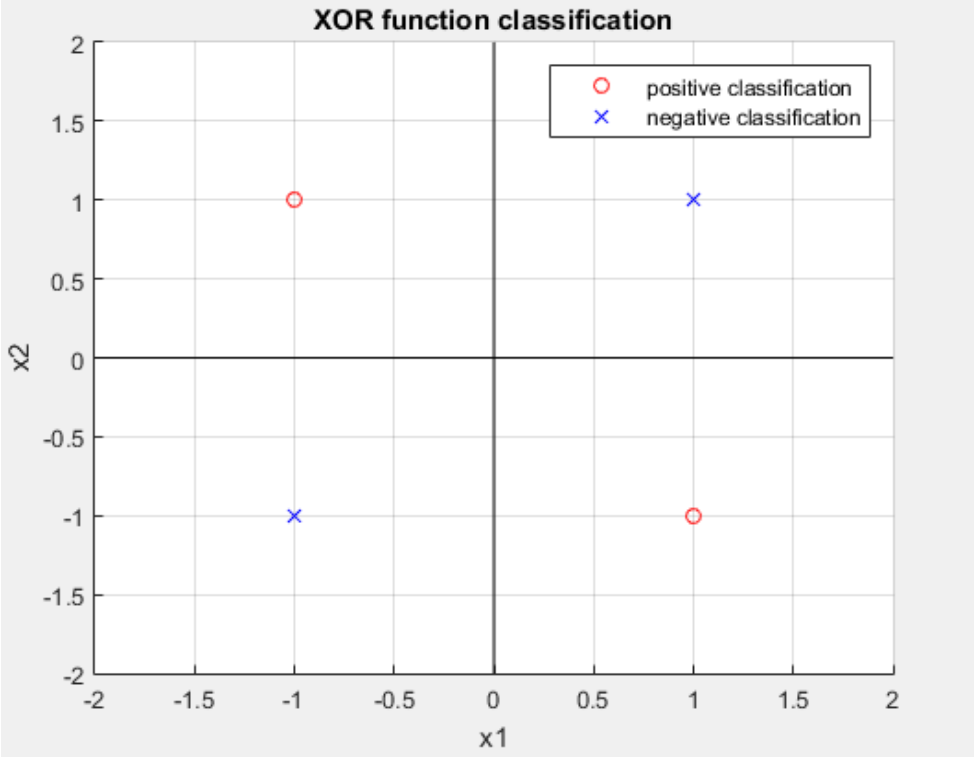

Figure 2 – XOR classification in input space.

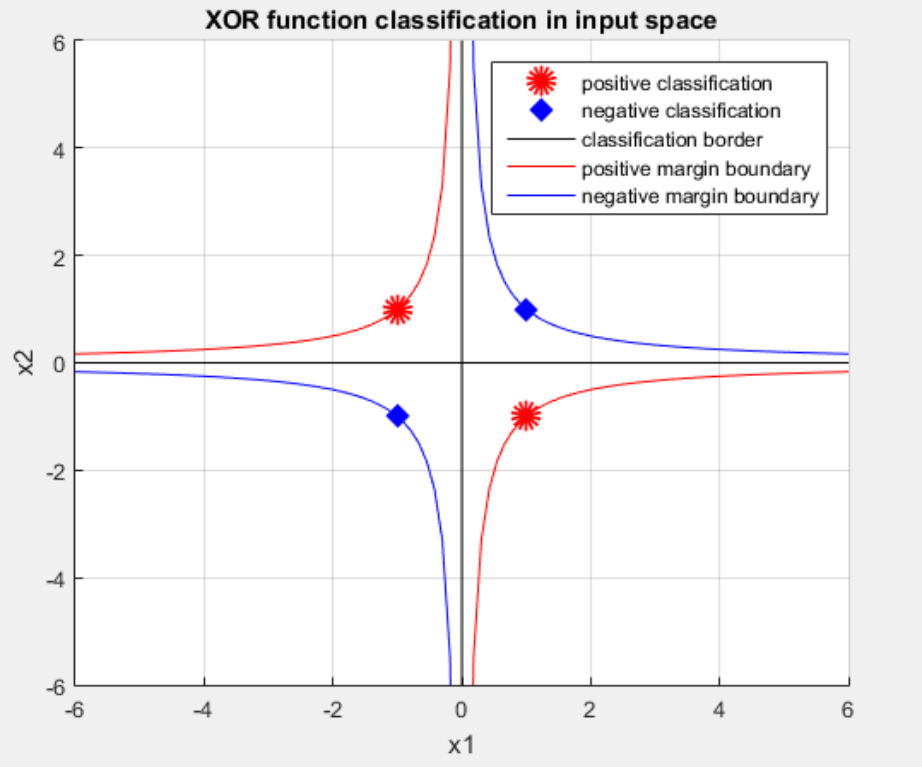

Figure 2a – XOR classification using higher dimension transformation.

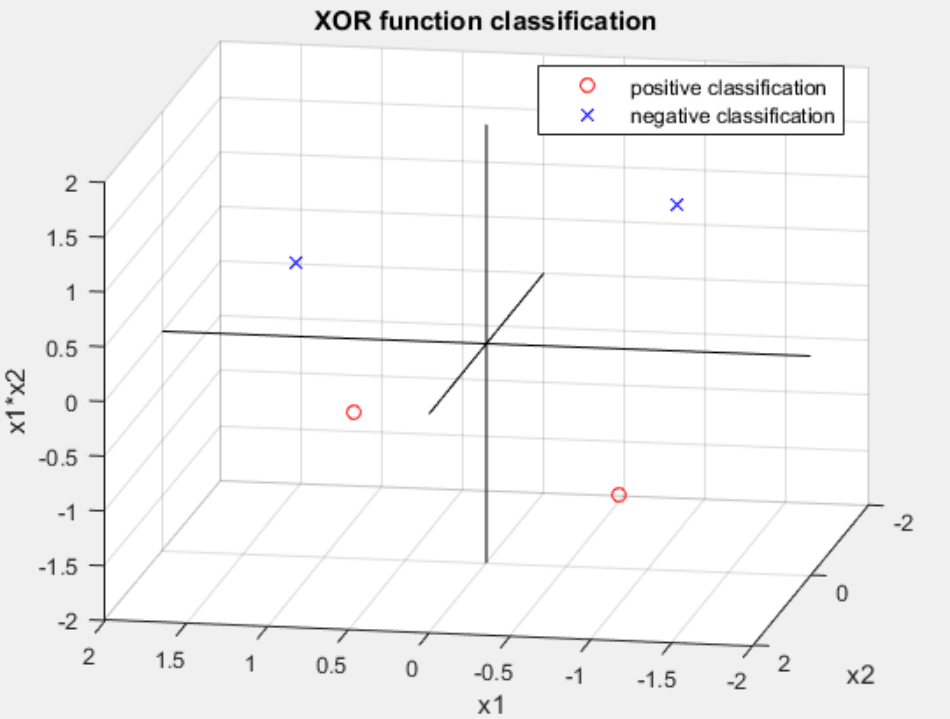

Figure 3 – XOR classification in feature space.

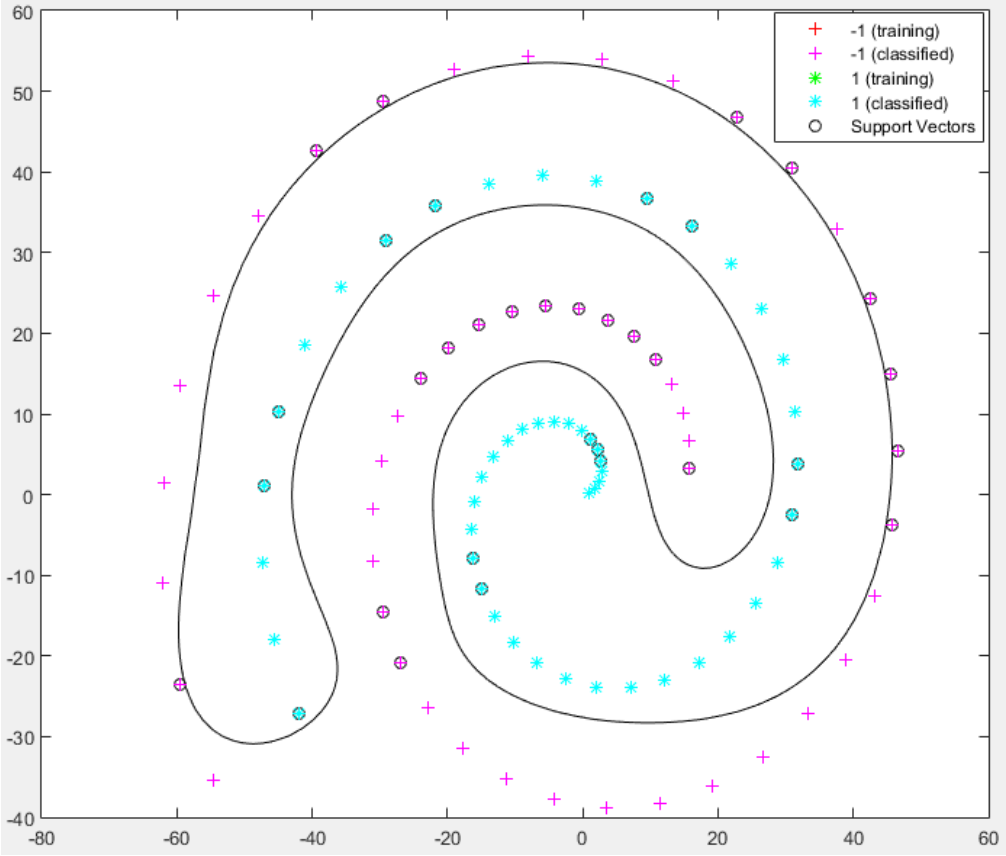

Figure 4 – Spiral classification using polynomial kernel of power 6.

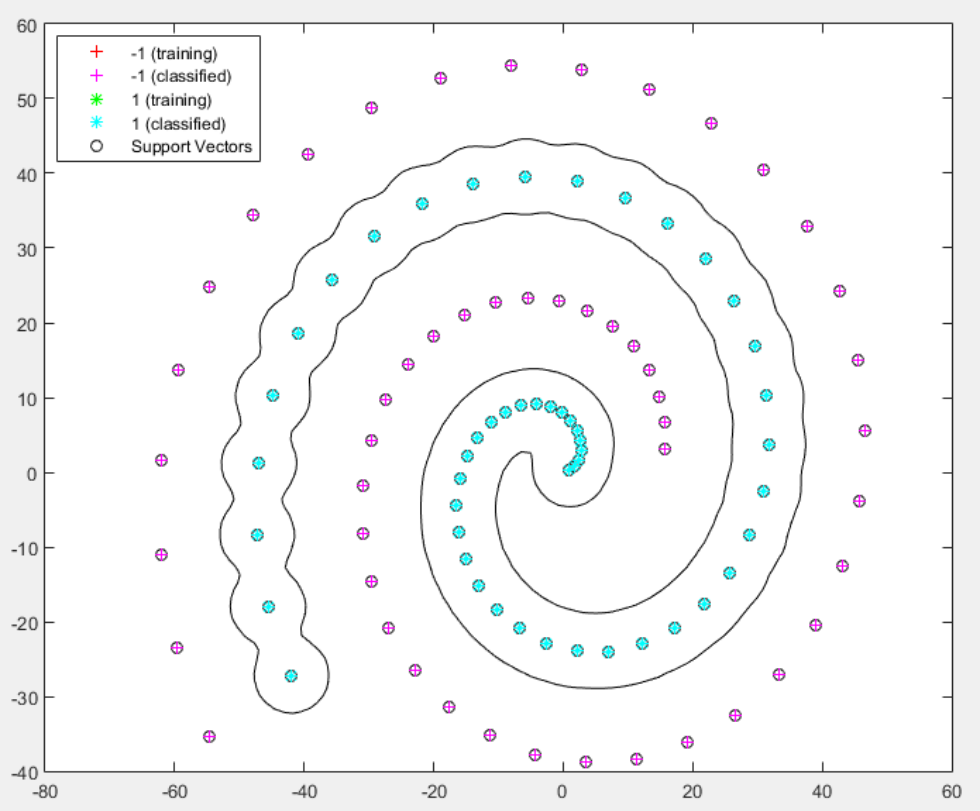

Figure 5 – Spiral Gaussian RBF classification with σ = 0.1.

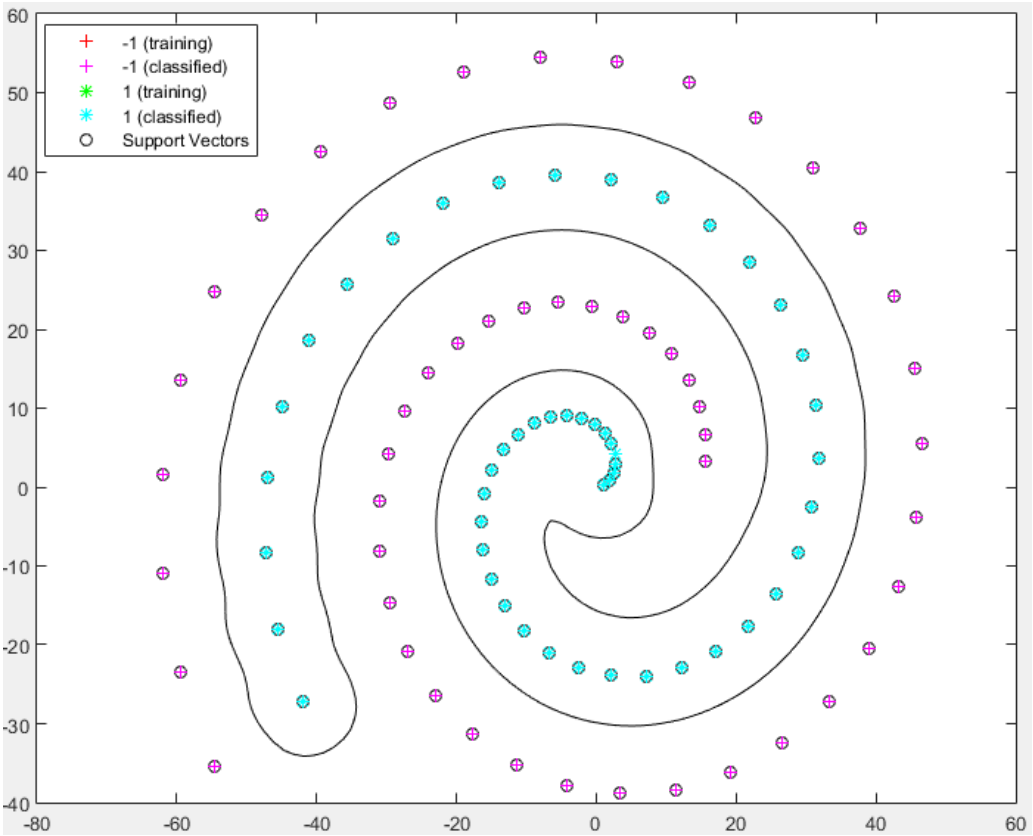

Figure 6 – Spiral Gaussian RBF classification with σ = 0.15.

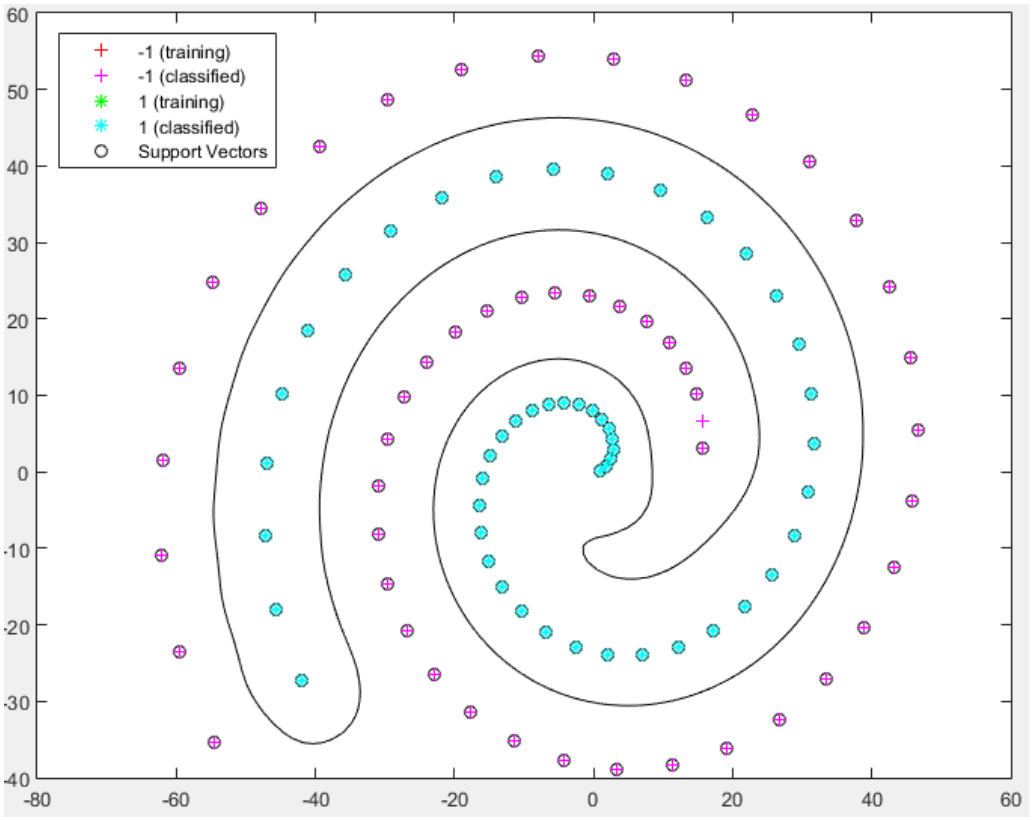

Figure 7 – Spiral Gaussian RBF classification with σ = 0.2.

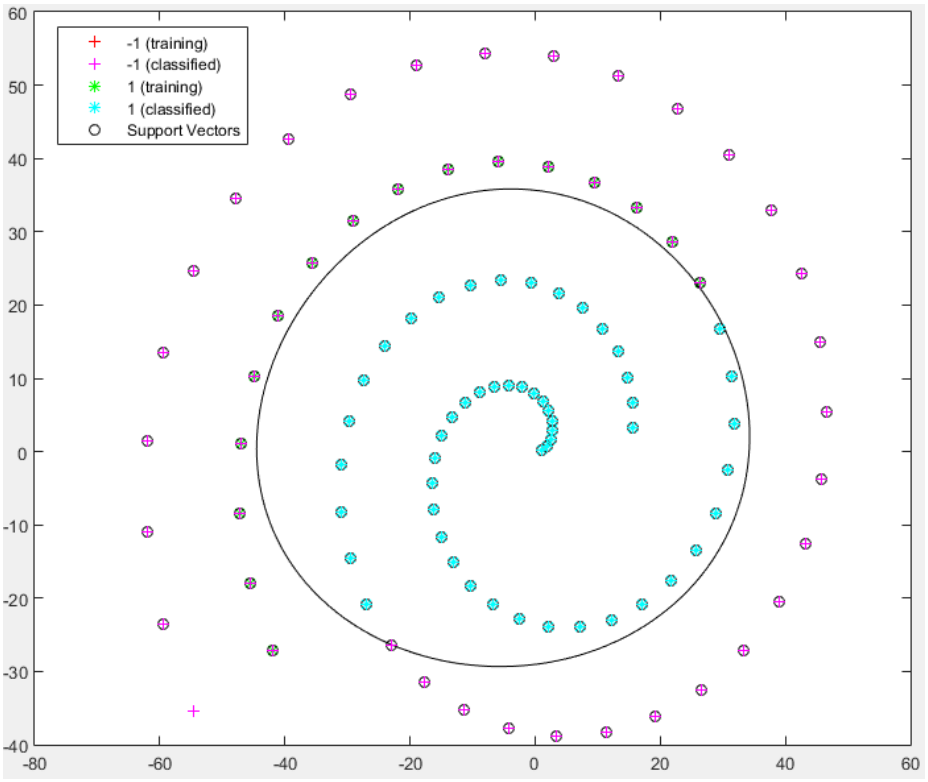

Figure 8 – Spiral Gaussian RBF classification with σ = 10.

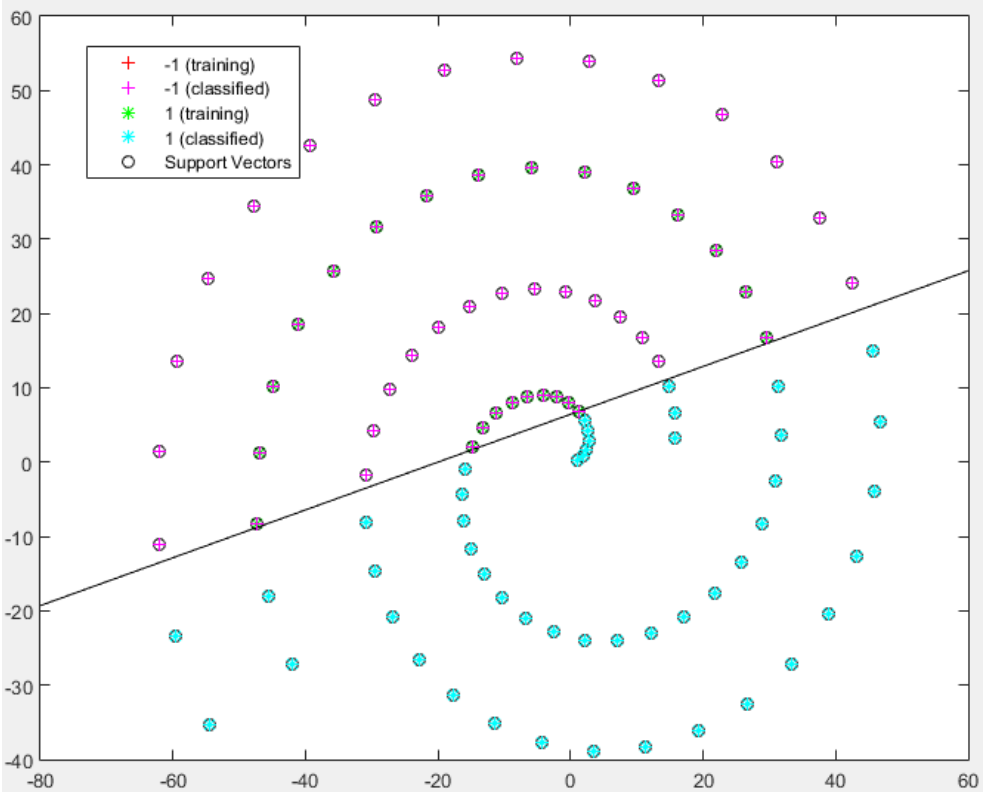

Figure 9 – Spiral Gaussian RBF classification with σ = 1000.

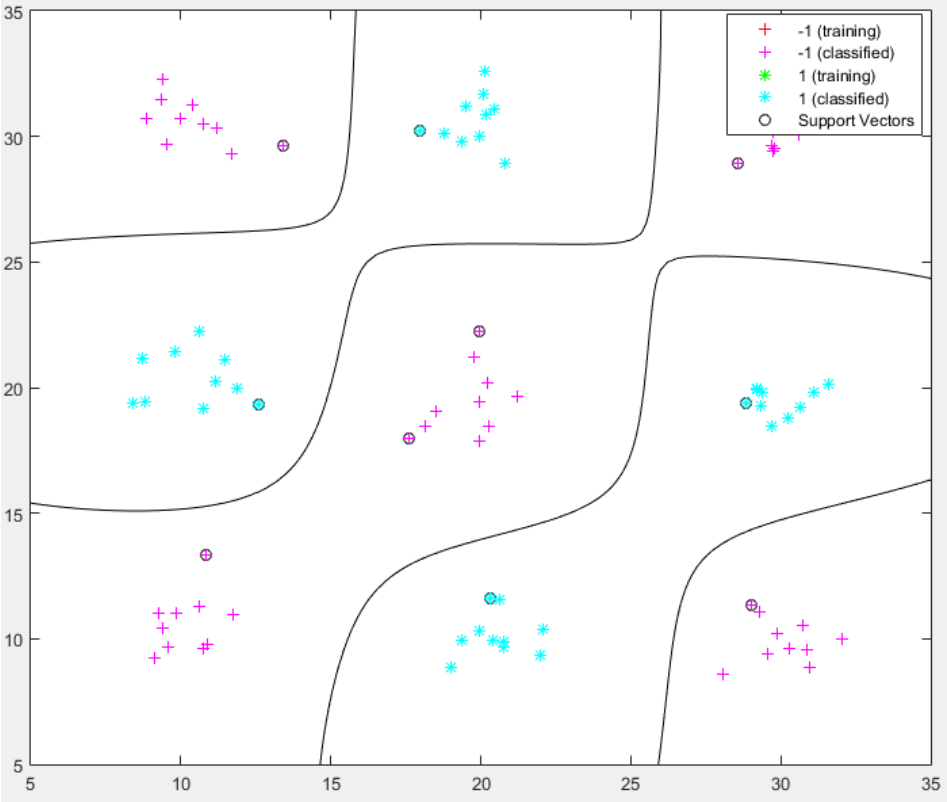

Figura 10 – Gaussian RBF classification with σ = 1.

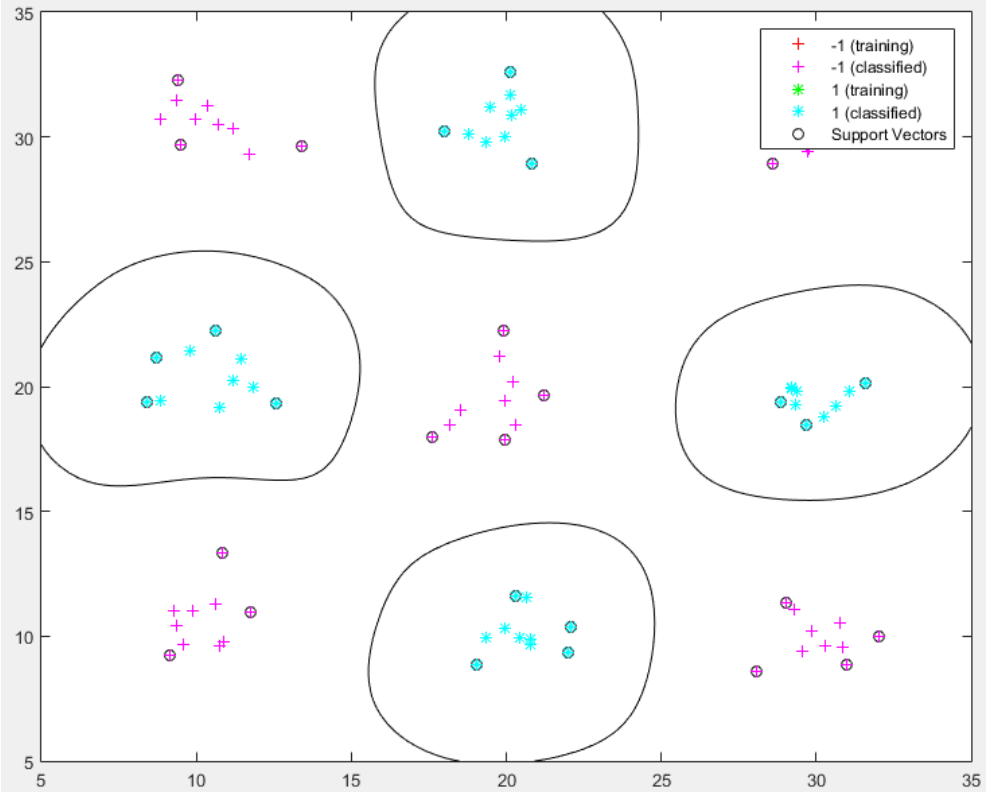

Figura 11 – Gaussian RBF classification with σ = 0.25.

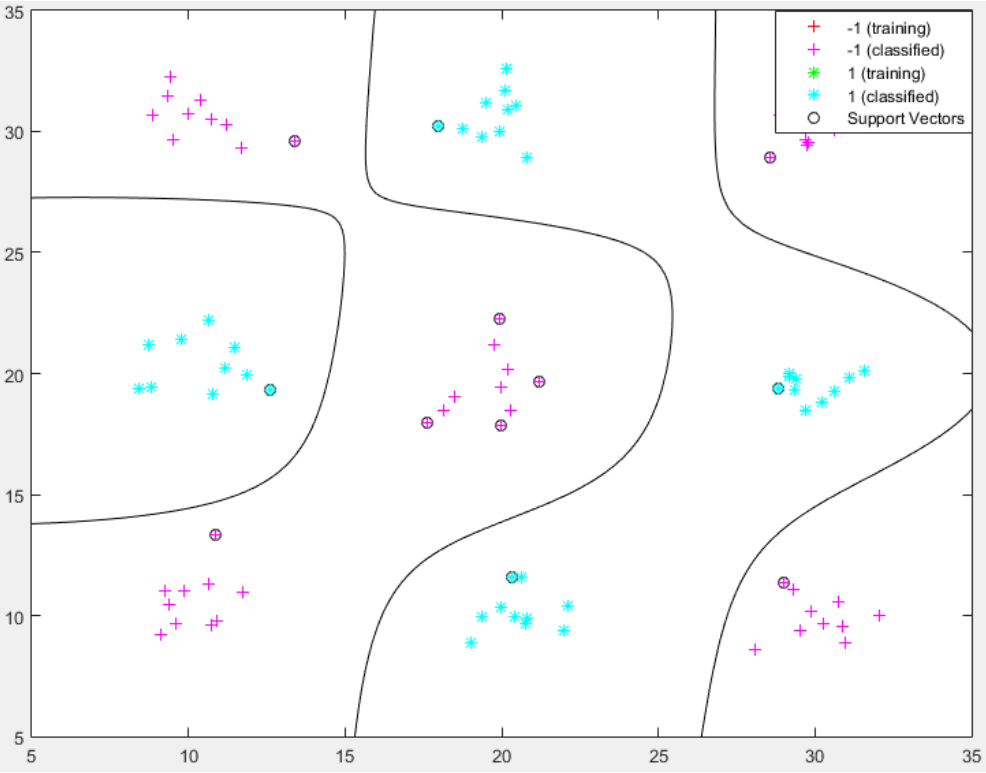

Figura 12 – Gaussian RBF classification with σ = 2.5.

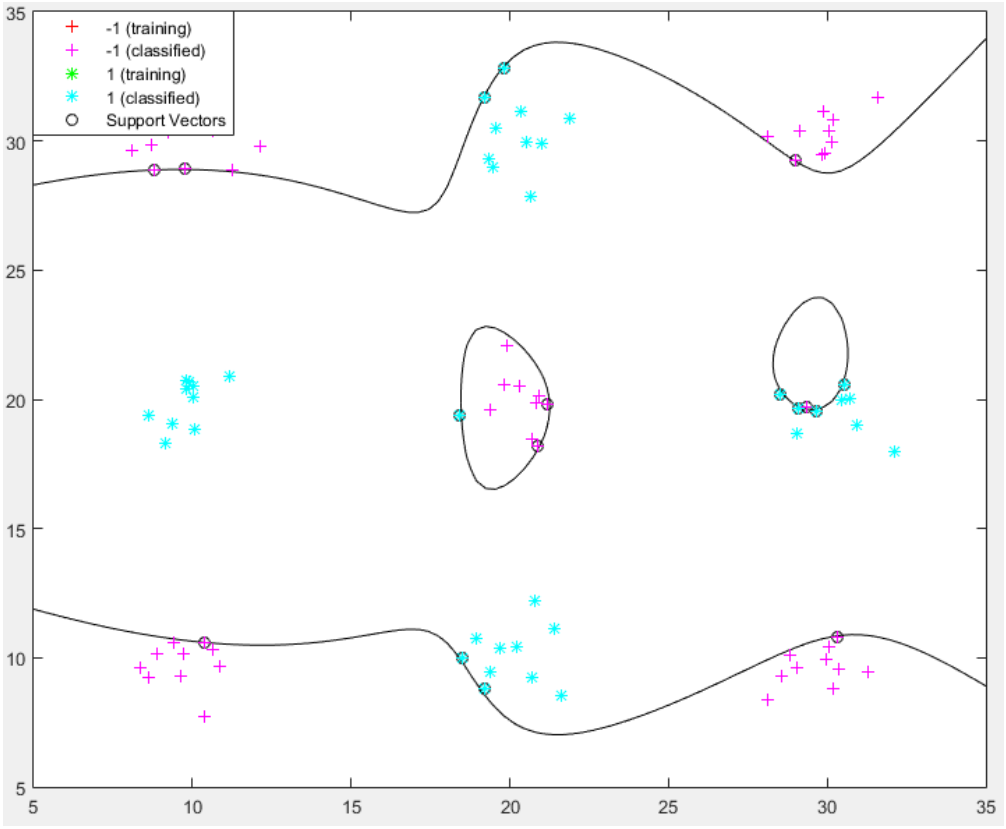

Figura 13 – Hard margin Gaussian RBF classification with σ = 1 (outliers present).

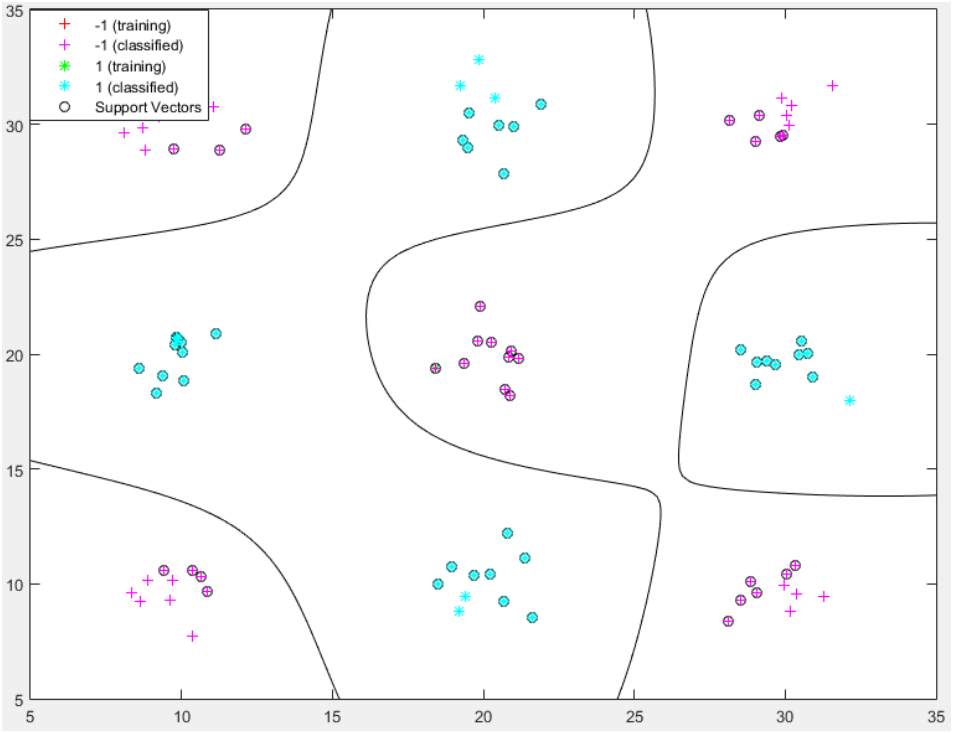

Figura 14 – Soft margin Gaussian RBF classification with σ = 1 and constraint 10(outliers present).

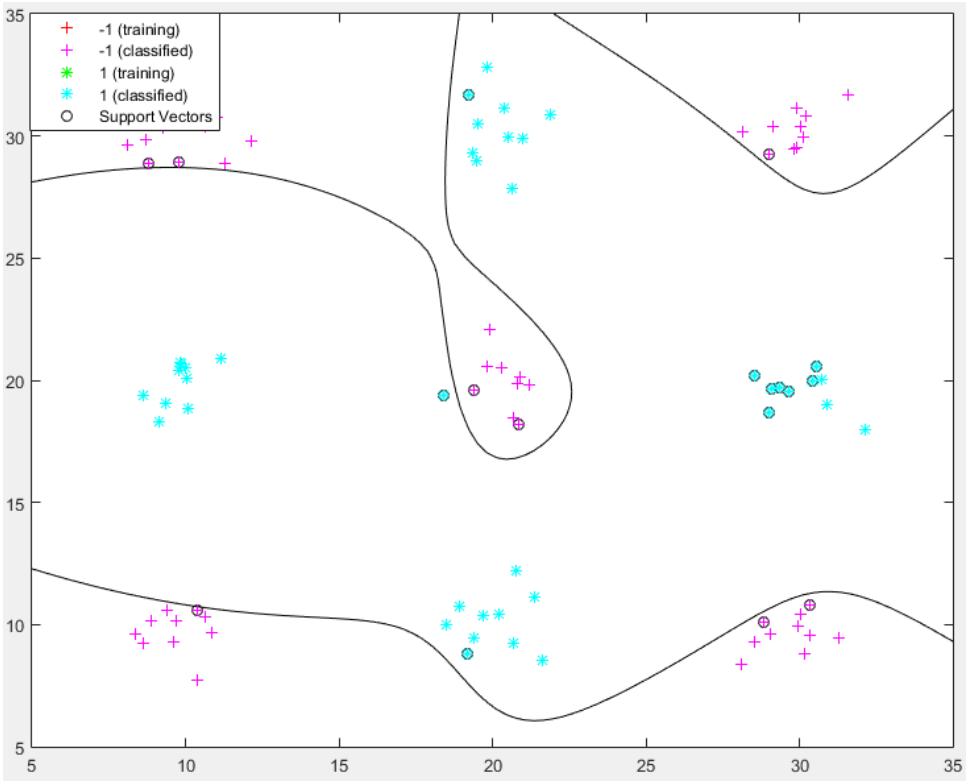

Figura 15 – Soft margin Gaussian RBF classification with σ = 1 and constraint 10000 (outliers present).

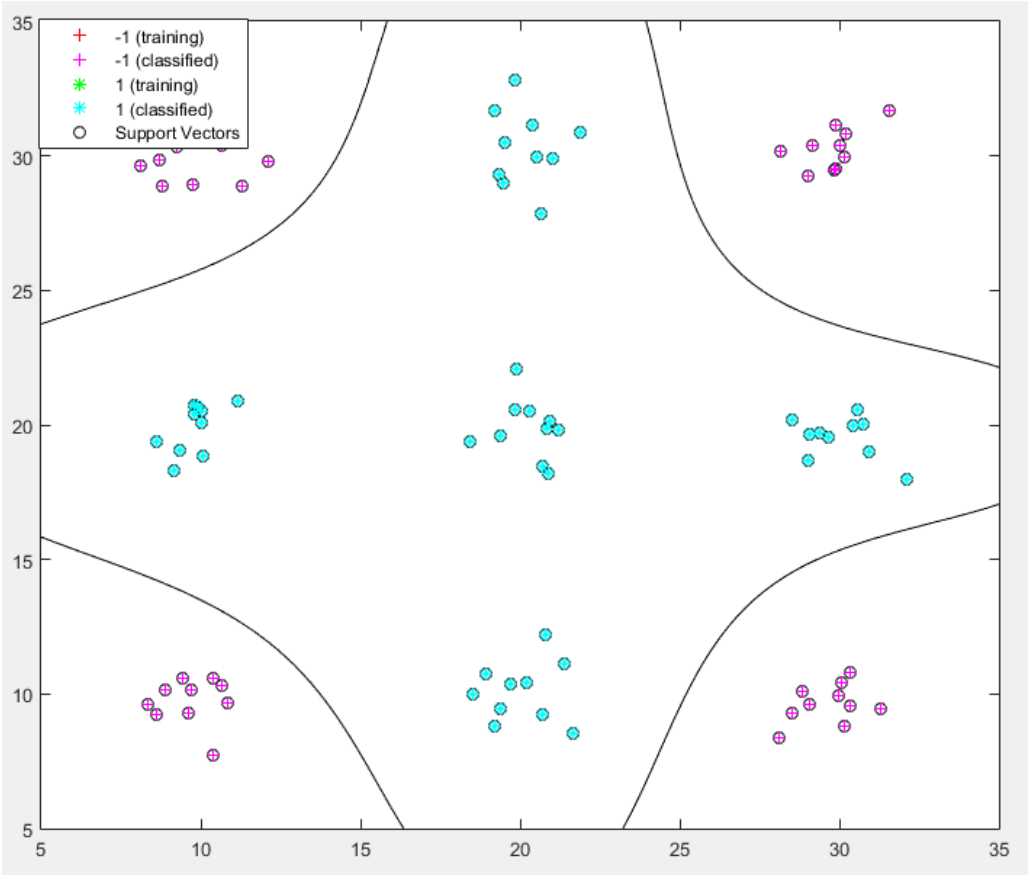

Figura 16 – Soft margin Gaussian RBF classification with σ = 1 and constraint 0.01 (outliers present).