Relatório LAB 5 AA – SVM (depois passamos para PDF)

2.1

By inspection we can find the following elements:

Maximum-margin separating straight line:

Support vector for class 1:

Support vectors for class -1:

Margin boundary for class 1:

Margin boundary for class -1:

(Ver resolução do sistema de equações no caderno:)

2.2

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.

(Ver cálculos e referencial 3D no caderno)

2.3

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Class |
| -1 | -1 | 1 | -1 |
| -1 | 1 | -1 | 1 |
| 1 | -1 | -1 | 1 |
| 1 | 1 | 1 | -1 |

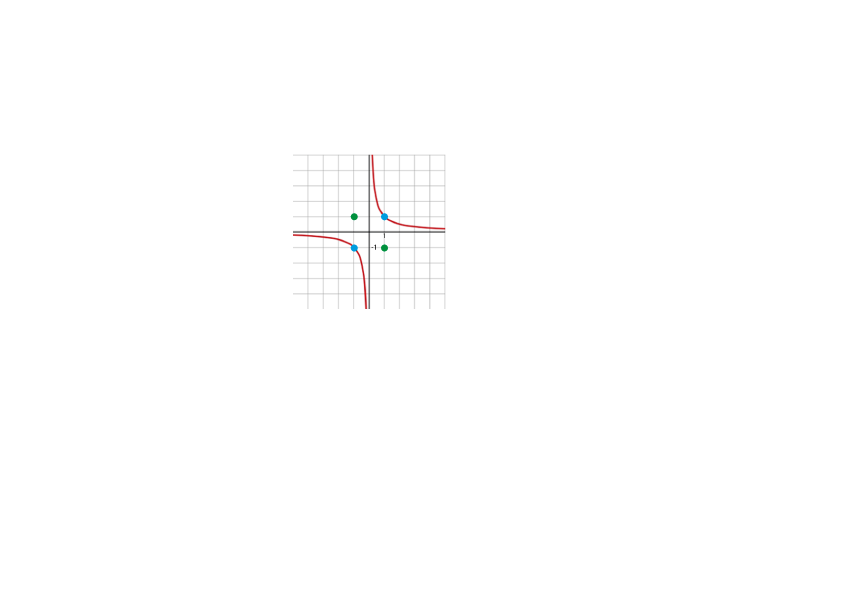
After computing the elements associated to the XOR function we can observe that we will have all elements from class -1 in a horizontal plane with a height of 1 and that all the elements from class 1 will be in a horizontal plane of height -1. This means that all vectors are support vectors.

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

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

b = -2

2.4

Classification border:

=> Class -1

=> Class 1

Margin boundaries:

For class -1:

For class 1:

(Ver plot no caderno)

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

3.1

This kernel corresponds to a mapping from a 2-D space into a 3-D space. The higher space dimension is determined by “p”. 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:

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

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

So from this it follows that the feature map is:

3.2

4.1 – Testing SVM with Poly kernel and finding optimal polynomial order

|  |  |  |
| --- | --- | --- |
| 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 - Testing SVM with RBF kernel and sigma value

|  |  |  |
| --- | --- | --- |
| 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 . We also conclude that for extreme values of sigma () the SVM overfits the data drawing boundaries around each example as we see in Figure 1. In conclusion the optimal value for sigma is in the interval . This interval yields good performance and avoids overfitting

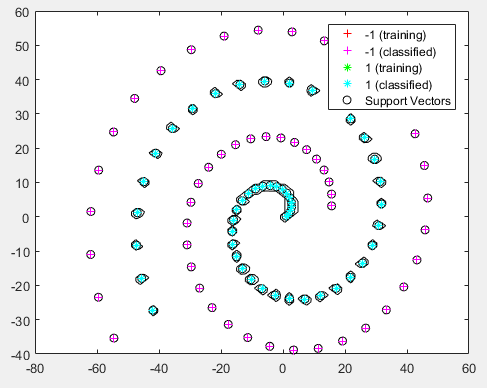


Figure 1 - Testing SVM with RBF kernel with sigma = 0.01

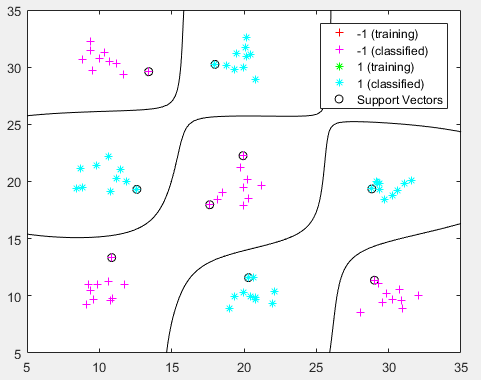
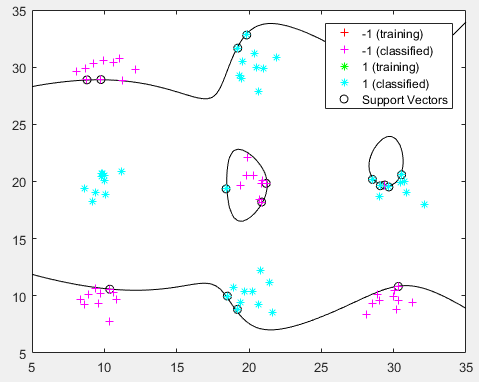
4.3 - Testing SVM with RBF kernel and sigma value w/boxconstraint

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

4.4

We used 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 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, one that clearly overfits that data.

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



4.5

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

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

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.

Figure below shows different decision boundaries for each *boxconstraint* value. On the left its 10^1 and on the right 10^7. We can see that using *boxconstraint = 10^1* the decision boundary approximates the one obtained when not using *boxconstraint*.

