Instituto Superior Técnico

Departamento de Engenharia Electrotécnica e de Computadores

Machine Learning

5th Lab Assignment

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.

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.

[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.

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.

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.

3 Practical Assignment

`Note1` In this assignment, you'll often find between brackets, as in {`command`}, suggestions of Python commands that may be useful to perform the requested tasks. You should search Python documentation, when necessary, to obtain a description of how to use these commands.

`Note2` We suggest the use of function `plot_contours`, which was provided, to visualize the results of your experiments.

When using SVM, you are required to specify the kernel function. For a linear classifier you should choose the linear kernel (*i.e.,* the feature space is equal to the input space),

For nonlinear classifiers there are a number of kernel options. One which is commonly used in pattern recognition problems is the Gaussian or radial basis function (rbf) kernel

$$K(\mathbf{x}, \mathbf{y}) = e^{-\gamma \|\mathbf{x}-\mathbf{y}\|^2}, \tag{5}$$

where γ is a scale parameter.

This kernel has several useful properties, as it is shift-invariant and isotropic.

Another commonly used kernel is the polynomial one, defined by

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + a)^p, \tag{6}$$

1. (a) Load the files `spiral_x.mat` and `spiral_y.mat` which contain the classical spiral example, with 50 patterns per class.{`load from numpy`}
   (b) Instantiate an SVM classification model with a polynomial kernel. Set the maximum number of iterations to 100000. Fit the SVM model to the spiral data. {`SVC and fit from sklearn.svm`}
   (c) Determine experimentally, using the polynomial kernel, the value of p for which you get the best classifier (start with $p = 1$) on your training data. Write down all the experiments performed, together with the classification error percentages and number of support vectors. Comment on the results you obtained. {`acc from sklearn.metrics and plot_contours`}
   (d) Using the same data try now the Gaussian RBF kernel. Find the approximate value of γ for which you can get the best classifier. Comment on the effect of γ on your results.

2. (a) Load the files `chess33_x.npy` and `chess33_y.npy` which contain 90 patterns per class arranged in a chess pattern.
   (b) Use a Gaussian RBF kernel and set 'C' parameter to Inf to enforce a hard margin SVM, for separable data.
   (c) Find a value of γ that approximately minimizes the number of support vectors, while correctly classifying all patterns. Indicate the value of γ and the number of support vectors.

3. (a) Load the files `chess33n_x.npy` and `chess33n_y.npy` which contain data similar to the one used in the previous question, except for the presence of a couple of outlier patterns.
   (b) Run the classification algorithm on these data with the same value of γ, and comment on how the results changed, including the shape of the classification border, the margin size and the number of support vectors.
   (c) Now reduce the value of 'C' parameter in order to obtain the so-called *soft margin* SVM. Try different values (suggestion: use powers of 10). Comment on the results.