Exercice 7 : Support vector machine

marie.bruguet

October 2021

Introduction

In this exercise, the goal is to generate a binary response variable, and then train some SVM on that data. The values of the parameters in the estimations will vary and effect the predicted classes.

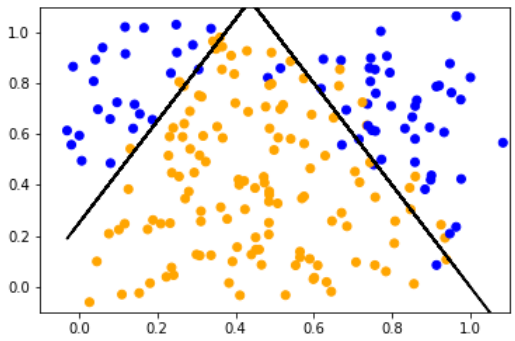
1 Generate data and visualisation (Q1 to Q4)

The first step is to generate some random data from an uniform distribution for 200 observations.
Two samples are generate (x_1 and x_2) and labels are attribute, following the rule :

$$\begin{cases} \text{orange} & \text{if } 2x_1 + 0.25 - x_2 > 0 \text{ and } -2x_1 + 2 - x_2 > 0 \\ \text{blue} & \text{otherwise} \end{cases}$$

Then we add noise to each observations, following a random normal distribution.
Finally, the observations are plot in $x_1; x_2$ scatter plot and the *true boundaries* are also plot. Since some noise has been added, the boundary doesn't perfectly separated the observations.

Figure 1: True boundaries



2 Train a Support Vector Machine (SVM)

Before training the SVM, the label are transform to dummy to be able to become predictable. Orange becomes 1 and Blue becomes 0.
Moreover, the dataset is split in train (80%) and test (20%) with X the matrix of the 200 observations built with the (x_1 and x_2) couple and y the vector that contain the real attributed color label for each of the 200 observations.

2.1 Using a linear Kernel (Q5)

The SVC function from the sklearn.svm librairy is use to train the data.
There are several different hyperparamter to define. For this first train, only the *kernel* one is define and put to *linear*. By default, there will be :

- C (regularization parameter) : 1.0
- gamma : 'scale'

Table 1: Classification report for the SVM train and prediction with a linear kernel

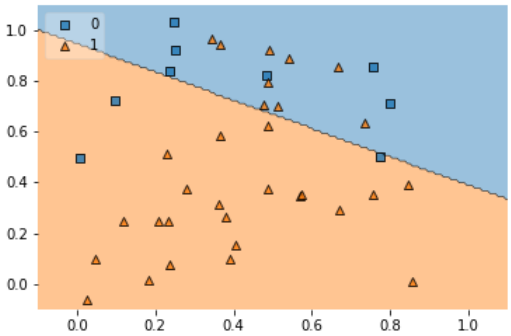
	Precision	Recall	Support
blue	0.40	0.67	9
orange	0.88	0.71	31
		Accuracy	0.70

The precision allows to know how many observation where actually well predict. Here 40% of the observations predicted as blue were also blue in the real dataset and 88% of the observations predicted as orange were also orange in the real dataset.
The difference of percent of good precision can be in part explain by the fact that there are more orange observations then blue one, thus the algorithm has more data to train on for the orange observations.

The recall allows to compute the percentage of good prediction of a class, regarding the number of real observation available in this same class.
Here, for exemple for the blue class the recall is 67%, it means that the algorithm were able to predict 67% of the real blue observation that exist in the dataset.

The classification accuracy is the ration of number of correct predictions to the total number of input sample. It means that for the 40 test observations furnish, the algorithm predicted well 70% of them, so 28 observations.

Figure 2: SVM - linear kernel



Since a linear kernel is use, there is only one boundary to separate the two classes. It is obvious that this is not the best configuration to train the SVM since there are blue observations in the orange predict part and they are far from the boundary.

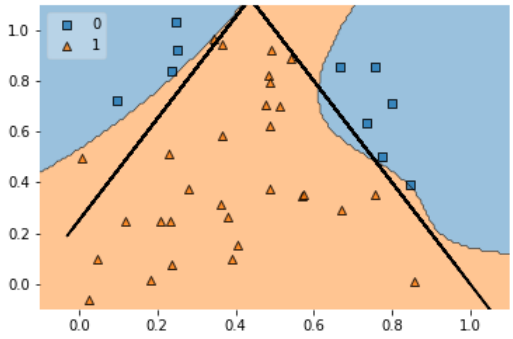
2.2 Using a polynomial kernel (Q6 to Q9)

2.2.1 Choice of the degree

This time the kernel use is polynomial and the goal is to find the best degree of the polynom to have the best prediction.
Since the true boundaries are known, the idea is to visualize the differents boundaries built by the algorithm depending on the degree of the polynom and compare it with the true boundaries.
The choice is made to built a loop for different polynom going from 2 to 9. From looking at the plots, it seems that a degree of 5 can be a good hyperparameter for the algorithm.

Using the algorithm trained with a polynomial degree of 5, the predicted values look like on the following plot.

Figure 3: SVM - polynamial kernel with 5 degrees



Since we have the true boundaries draw on the plot, it is easy to see that there is one observation that is wrong predicted. However, it seems all in all to be a pretty good prediction. This intuition is confirmed by the classification report. The accuracy rate is 88% which is way better than with the linear kernel (70%) and both the precision and recall parameter have increase.

Table 2: Classification report for the SVM
train and prediction with a polynamial degree of 5

	Precision	Recall	Support
blue	0.70	0.78	9
orange	0.93	0.90	31
		Accuracy	0.88

2.2.2 Choice of the C parameter

In a Support Vector Machine there are two goals : find a hyperplane with the largest minimum margin, and a hyperplane that correctly separates as many instances as possible. The problem is that we will not always be able to get both things. The C parameter determines how great is the desire to have an hyperplane that correctly separates instances.
The choice is made to vary the C parameter from 0.1 to 1 with a step of 0.1. We can see on the different plots generated that there is one observation that is classify as *orange* or *blue* depending on the value of the C parameters. From our "true" data we know that it is supposed to be predict as *orange*.

Table 3: Impact of the value of the C hyperparameter on the label prediction

C	Predicted color
0.1	blue
0.2	orange
0.3	orange
0.4	orange
0.5	orange
0.6	blue
0.7	blue
0.7	blue
0.8	blue
0.9	blue
1	blue
10 and more	orange

Since the C parameter determines how great is the desire to have an hyperplane that correctly separates instances, once it is "high", the switching observation is finally predict as *orange* even if it may be reduce the minimum margin. Thus the choice of the C hyperparameter depends on how the scientist decide to model the algorithm and want is wanted as prediction.

3 Using a Gridsearch to model the best hyperparameters (Q10)

The goal is to find the hyperparameters C and the degree that will give the best classification prediction. This cannot be learned from the data, thus we have to test severals combination of hyperparameters and evaluate which one leads to the best prediction.

We use a Gridsearch to test several combination rather than do it by hand. The Scikit-Learn *GridSearchCV* function takes a dictionary that describes the parameters that could be tried on a model to train it. The grid of parameters is defined as a dictionary, where the keys are the parameters and the values are the settings to be tested.

A dictionnary of hyperparameters to test is built and is as follow :

Table 4: Choice of grid parameters

Hyperparameters	Values
C	[0.1, 1, 10, 100]
degree	[0.5,1,2,3,4,5,6,7]
gamma	scale
kernel	poly

Moreover, when calling the Gridsearch function, the verbose number has to be choose. It correspond of the number of kfolds in the cross-validation process.
Indeed, the fit from the Gridsearch work as follow : it runs the same loop with cross-validation, to find the best parameter combination. Once it has the best combination, it runs fit again on all data passed to fit (without cross-validation), to built a single new model using the best parameter setting.
The choice is here made to have 3 kfold in the cross-validation, as it is ussualy the case. Finally, the algorithm will have to compute $(8 * 4) = 32$ possible combinaison of hyperparameters and run $(32 * 3) = 96$ models to compute the accuracy for each one on three different parts of the train sample.

From the report of classification prediction, it seems that this model is less accurate that the one from the part 2.2.1 where the accuracy was 88% with also better precision and recall percent. However, the question is place in a situation where the true boundaries are unknown. Thus having an accuracy rate of 85% is acceptble. We still have better predictions for the orange class, with a precision rate of 83% i.e. 83% of the observations predict as orange are well orange in the real dataset. For the blue prediction the precision rate is only at 64%.

Table 5: *Best* model

Hyperparameters	Values
C	0.1
degree	7
gamma	scale
kernel	poly

Table 6: Classification report for the SVM train and prediction with the *best* model

	Precision	Recall	Support
blue	0.64	0.78	9
orange	0.83	0.87	31
		Accuracy	0.85

Figure 4: SVM - Polynomial kernel with *best* model