Module-Lab 2: Probabilistic Machine Learning

Exercise 1 This Exercise is part of the Lab/Practical session of the ACP ("Aprendizagem Computacional Probabilistica"). Please, open the compressed folder <available in UC-Student> where you will find the following Dataset's files (in .txt format):
X1.txt, X2.txt, Labels.txt

Each line in the file/dataset corresponds to an examples/entity/object thus, the dataset contains 1400 examples. Students can use Matlab or Python to coding. In this exercise X1 comprises the outputs of a given supervised classifier (eg, a Linear Discriminant), X2 provides the output of a generative classifier (eg, a Naïve-Bayes), finally the labels/**ground-truth** is given by the variable named *Labels*. This is a two/binary classification problem.

Let's consider **positives** all the examples labelled by 1 (ie, digit = 1) while label = 0 represents the negative class.

To be able to make decisions a classifier depends on a decision-threshold ***(thr)***. Because this is a 2-class problem, it is common to assume that the classifier's prediction is 'positive' when the output is greater than a *threshold* ***(thr)***. If the output$_{\mathbf{X1}}$ is ≥ zero (ie, **thr=0.0**) than we say the classifier decision is a 'positive' otherwise the example is 'negative'. On the other hand, for output$_{\mathbf{X2}}$ we have to consider **thr=0.5**.

a) So, considering ***thr* = 0.0** for the 1st classifier (ie, "X1") and ***thr* = 0.5** for the classifier "X2", develop a code to compute the following performance measure per classifier:

- The number of examples per class
- The number of true positives (TP) per class
- The number of true negatives (TN) per class
- The number of false positives (FP) per class
- The number of false positives (FN) per class
- The respective rates ie, TPrate, TNrate, FPrate, FNrate

b) Using the same code developed above or by developing functions calculate:

- Accuracy
- Balanced accuracy
- F1/F-score
- Precision
- Recall

c) We may have noticed that the number of 'negatives' surpass the positives. Now, modify the first 400th values of X1 to **-1** (ie, let us now artificially force all the positives to error). Then, calculate the resulting Accuracy and the Balanced Accuracy. Compare the achieved values against the original/unmodified values.

➔ You may have noticed that X1 is not in the interval [0,1] ie, it is not normalized.

d) Perform a normalization on X1, using a monotonic function (eg, **Sigmoid**), in order that its values now belong to the closed interval [0,1]. Using the same code above (b), but this time consider **thr = 0.5** for the classifier "X1", calculate the same performance measures in (b). The final performance for X1 should be the same or, if not, very close eventually.

e) Plot a histogram of the X1 values that correspond to the 'positives' then, in the same graph but using different color, plot the values of X1 that correspond to the 'negative'. You will be able to see the 'valley' between both histograms and thus the 'optimal' separating value – which is 0.5, isn't ?.

Exercise 2 (combining classifiers)

Now, let's study basic late-fusion rules used to combine the output of both classifiers.
a) Assuming X1 has been normalized, develop a code to calculate the following combining rules:

- Average
- Minimum
- Maximum

b) Compare the classification performance, using Balanced accuracy, of the single classifiers vs the 3 combining rules above.

Exercise 3 (Receiver Operating Characteristic curve - ROC)

The ROC is useful to visualize the performance of a classifier for different threshold values. Among other things, it allows us to calculate the AUC (Area Under Curve) which is another useful performance measure. ROC is a graph where TPR (True Positive Rate) is plotted on the y-axis *versus* the FPR (on the x-axis). Each pair of points in the curve (TPR_i, FPR_i where i=1,…,n) is plotted by varying monotonically the threshold value.

a) Develop a code/function to calculate TPr and FPr, for the X1.'normalized', using the following threshold values: 0.0, 0.05, 0.1, 0.15, …, 0.95, 1.00. So, the final ROC curve will have n=21 points.

b) Repeat (3.a) but this time for X2.

c) Calculate the AUC. In Matlab you can use the function *trapz* (ie, trapezoidal integration).