E9 spec sheet:

confusionMatrix

Mission command preamble: As in general, we won't tell you *how* to do something. That is up to you and your creative problem solving skills. However, we will tell you what we would like this function to do. So here are the specs of the function we would like you to write.

Purpose: It is imperative to construct a confusion matrix from data in order to draw a ROC curve and find the area under the curve (AUC). This figure of merit can be used to characterize the performance of a classification algorithm. This topic can be rather confusing, so it is best to implement this in code. I recognize that it is late in the semester and the last thing you need is a long and extensive coding assignment, so all I'm asking for here is the actual confusion matrix, nothing more.

Specific things we would like this function to do:

a) Take in 4 input arguments, in this order: 1) An "x-base" – you can think of this as the value of some predictor. 2) The null distribution - how likely the corresponding x-base value is, if the signal is not present (say the customer is not buying the product). 3) The signal distribution – how likely the corresponding x-base value is, if the signal is present (say the customer is buying the product). 4) A threshold value x at which we cut the distributions to calculate the confusion matrix for that x-value.

b) The function should compute the full 2x2 confusion matrix: True positives, false positives, false negatives and true negatives.

c) The function should return the values computed in b) as a single 2x2 numpy array in the following arrangement: Upper left - true positives, upper right – false positives, lower left – false negatives, lower right – true negatives.

d) Assumptions: Input arguments should be arrays of arbitrary – but same, across the 3 variables – length. The last argument should be a real number within the range of the 1st input argument – the x-base.

e) Make sure the function has a clear header as to what inputs the function assumes, what outputs it produces and when it was written.

Hints

(you can do whatever you want, and there is an infinite number of ways to do this that are all correct and valid, but some people find some suggestions helpful, so here they are. But this is just one possible approach):

1) Find the index value of x (the first input argument) that corresponds to the threshold value (the 4th argument). What is the location in x where the value of x corresponds to the threshold?
2) Calculate the true positives as the sum of all values in the signal distribution (the 3rd input argument) from the index value calculated in 1), plus 1 (the next one) to the end of that distribution.
3) Calculate the false positives as the sum of all values in the null distribution (the 2nd input argument) from the index value calculated in 1), plus 1 (the next one) to the end of that distribution.
4) Calculate the false negatives as the sum of all values in the signal distribution (the 3rd input argument) from the 1st index value to the index that corresponds to the threshold (calculated in 1)), inclusive.
5) Calculate the true negatives as the sum of all values in the null distribution (the 2nd input argument) from the 1st index value to the index that corresponds to the threshold (calculated in 1)), inclusive.
6) The reason we have to sum up to the index that represents the threshold value vs. summing the values larger than that in 2) to 5) is that we should only count the value that corresponds to that index once, and conceptually it makes sense to think of values exceeding the threshold to be considered as "present"
7) Arrange the values calculated in 2) to 5) in the confusionMatrix laid out in c) and return it.

Input / output examples:

x = first column of sampleConfusion1.csv

y0 = 2nd column of sampleConfusion1.csv

y1 = 3rd column of sampleConfusion1.csv

Input: confusionMatrix(x,y0,y1,2)

Output:

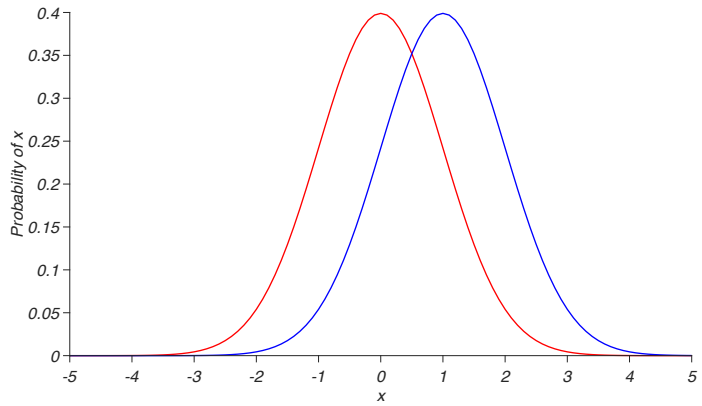
1.467	0.201
8.532	9.799

Input: confusionMatrix(x,y0,y1,1.5)

Output:

2.911	0.605
7.089	9.395

Below is a visualization of the situation (the null distribution in red vs. the signal distribution in blue, as a function of the predictor x), just so you can see what is going on. Your function doesn't have to do this.



x = first column of sampleConfusion2.csv

y0 = 2nd column of sampleConfusion2.csv

y1 = 3rd column of sampleConfusion2.csv

Input: confusionMatrix(x,y0,y1,0)

Output:

9.735	0.193
0.265	9.807

Input: confusionMatrix(x,y0,y1,1)

Output:

7.259	0.003
2.741	9.997

Below is a visualization of the situation (the null distribution in red vs. the signal distribution in blue, as a function of the predictor x), just so you can see what is going on. Your function doesn't have to do this.