Northeastern University, Khoury College of Computer Science

CS 6220 Data Mining — Assignment 6

Due: November 14, 2023 (100 points)

So Man Amanda Au-Yeung
https://github.com/amandaay/CS6220DataMining/tree/main

Evaluating Your Classifier Performance

In lecture, you will recall that you can calculate the ROC curve by computing the true positive rate (TPR = Probability of Detection, i.e. P_D) and false positive rate (FPR = Probability of False Alarm, i.e P_{FA}) at every threshold. In this homework, our function will take in predicted classifier scores (scores from Livermore Laboratory sensors) and the true labels (labels of whether or not laser tubes are calibrated).

ROC/AUC from min-FPR to max-FPR

We will implement a special function that efficiently plots the ROC curve and calculates the AUC for a specified range of FPR. Where your function differs from traditional ROC curve calculations is that it will compute only the true positive rate (i.e., TPR, or P_D) from the range starting from some minimum FPR (i.e., min-FPR, or min-$P_{F\,A}$, defaulted to 0) to some specified maximum FPR (i.e., max-FPR, or max-P_{FA}, required parameter).

In particular, it will then return all the FPR values and TPR values that can create a ROC curve in the specified FPR range, as well as the area under it (AUC). For example, I can specify max-FPR = 0.4, and our function will give us all the ROC scores in the range FPR = [0, 0.4], inclusive of the end values. While there may be a way to leverage existing libraries (e.g., Scikit- Learn), please refrain from doing so (though you can feel free to check your answers with them.)

The function signature looks like the following:

```
def roc_range(scores, labels, maxfpr, minfpr = 0)
  '''
```

[redacted]

1

```
  Inputs:
    scores - predictions from any given classifier
    labels - the true label (either 0, 1) of the data
    maxfpr - the maximum false positive rate (FPR)
    minfpr (optional, default = 0) - the minimum false positive rate (FPR)
  Outputs:
    fpr_in_range - list of FPR values from minpfa to maxpfa
    tpr_in_range - list of true positive rate (TPR) values from minfpr to maxfpr
    auc_in_range - single value of area under curve from minpfa to maxpfa
  '''
  return fpr_in_range, tpr_in_range, auc_in_range
```

Try it Out On Some Data

We will check your answers against our data, which you can find here at the course website. To read this data, feel free to use this code:

```
import numpy as np
data = np.load("assignment6.npz")
# The data that you will read in
scores_small = data['scores_small']
scores_large = data['scores_large']
labels_small = data['labels_small']
labels_large = data['labels_large']
```

This unpacks the data, which has some small test data that you can use to try out your algo- rithm before running the analysis on the larger set of data. If interested, this data has been drafted from National Ignition Facility readouts. Please make your code readable for any data with the above generic signature.

Homework Questions

1. Plot the ROC curve and calculate the AUC for the following ranges:

a) $P_{FA} \in [0, 1.0]$, the full range of thresholds

AUC = 0.82

b) $P_{FA} \in [0,0.4]$

AUC = 0.25

c) $P_{FA} \in [0,0.75]$

AUC = 0.58

d) $P_{FA} \in [0.25,0.75]$

AUC = 0.44

2. Your implementation notes:

a) Describe your implementation. How would you sweep your thresholds? For each threshold, how would you calculate the PFA and PD? What is the runtime in big-O notation?

My implementation starts using unique sorted scores as the threshold in descending order. Then converting each score to a binary prediction based on the threshold to calculate True Positive Rate (TPR or PD) and False Positive Rate (FPR). If exists within the threshold, then TPR and FPR are appended to the list. Then it's used to calculate the AUC. Runtime is O(N^2) with a nested for loop.

b) Determine the runtime of your implementation in big-O .

As mentioned in (a), the runtime of my implementation is O(N*2) due to the nested for loop.

c) Can you make your implementation run in O(N log N)?

Yes, we can sort the scores in descending order then iterate by calculating TPR and FPR. So we don't have to iterate through every threshold each time. Thus, we can sort (log N) then iterate the sorted scores (N), O(N*log N).

3. What thresholds provide a precision of 0.9?

Thresholds provide a precision of 0.9 are

[0.842, 0.843, 0.853, 0.855, 0.856, 0.858, 0.859, 0.862, 0.863, 0.865, 0.866, 0.867, 0.868, 0.869, 0.87, 0.872, 0.873, 0.874, 0.875, 0.878, 0.879, 0.881, 0.882, 0.883, 0.884, 0.885, 0.886, 0.887, 0.889, 0.89, 0.891, 0.892, 0.893, 0.894, 0.896, 0.897, 0.9, 0.901, 0.905, 0.908, 0.909, 0.912, 0.914, 0.917, 0.918, 0.919, 0.92, 0.921, 0.927, 0.928, 0.93, 0.931, 0.932, 0.933, 0.934, 0.937, 0.938, 0.939, 0.94, 0.942, 0.943, 0.944, 0.946, 0.947, 0.949, 0.954, 0.956, 0.958, 0.959, 0.96, 0.961, 0.962, 0.963, 0.964, 0.967, 0.968, 0.971, 0.972, 0.974, 0.976, 0.977, 0.978, 0.979, 0.981, 0.982, 0.983, 0.985, 0.988, 0.989, 0.99, 0.993, 0.994, 0.997, 0.999, 1.001, 1.002, 1.003, 1.004, 1.005, 1.008, 1.012, 1.016, 1.018, 1.02, 1.021, 1.026, 1.027, 1.029, 1.03, 1.031, 1.033, 1.035, 1.04, 1.041, 1.042, 1.046, 1.047, 1.048, 1.049, 1.059, 1.06, 1.061, 1.062, 1.064, 1.065, 1.067, 1.068, 1.07, 1.076, 1.078, 1.081, 1.082, 1.083, 1.084, 1.085, 1.086, 1.088, 1.091, 1.093, 1.094, 1.095, 1.096, 1.098, 1.1, 1.101, 1.102, 1.103, 1.104, 1.105, 1.106, 1.107, 1.108, 1.111, 1.116, 1.117, 1.119, 1.122, 1.124, 1.125, 1.126, 1.127, 1.128, 1.129, 1.131, 1.132, 1.134, 1.135, 1.136, 1.138, 1.14, 1.141, 1.143, 1.144, 1.146, 1.156, 1.161, 1.165, 1.167, 1.171, 1.173, 1.179, 1.184, 1.186, 1.188, 1.19, 1.191, 1.196, 1.197, 1.201, 1.206, 1.207, 1.211, 1.212, 1.214, 1.215, 1.216, 1.217, 1.218, 1.222, 1.223, 1.224, 1.225, 1.229, 1.232, 1.234, 1.235, 1.236, 1.239, 1.241, 1.244, 1.248, 1.249, 1.251, 1.254, 1.255, 1.256, 1.257, 1.258, 1.26, 1.261, 1.263, 1.266, 1.267, 1.268, 1.275, 1.276, 1.277, 1.279, 1.281, 1.288, 1.289, 1.291, 1.295, 1.305, 1.306, 1.307, 1.308, 1.309, 1.311, 1.314, 1.317, 1.318, 1.319, 1.321, 1.322, 1.324, 1.325, 1.329, 1.331, 1.332, 1.335, 1.338, 1.344, 1.345, 1.346, 1.347, 1.354, 1.365, 1.366, 1.368, 1.371, 1.372, 1.373, 1.374, 1.376, 1.377, 1.379, 1.38, 1.382, 1.383, 1.39, 1.391]

4. At this threshold, what is the accuracy of the classifier?

Accuracy of the classifier at this threshold

[0.5845, 0.585, 0.5855, 0.5855, 0.586, 0.586, 0.5865, 0.587, 0.5875, 0.588, 0.5885, 0.589, 0.589, 0.5895, 0.5895, 0.59, 0.5905, 0.591, 0.5915, 0.592, 0.5925, 0.593, 0.593, 0.593, 0.5935, 0.5935, 0.5935, 0.5935, 0.594, 0.594, 0.5945, 0.595, 0.5955, 0.596, 0.5965, 0.597, 0.5975, 0.598, 0.5985, 0.599, 0.5995, 0.5995, 0.6, 0.6, 0.6, 0.6005, 0.6005, 0.6005, 0.601, 0.601, 0.6015, 0.602, 0.6025, 0.603, 0.6035, 0.604, 0.604, 0.6045, 0.6045, 0.605, 0.6055, 0.606, 0.6065, 0.607, 0.6075, 0.608, 0.6085, 0.609, 0.6095, 0.61, 0.6105, 0.611, 0.6115, 0.6115, 0.612, 0.612, 0.6125, 0.613, 0.613, 0.6135, 0.6135, 0.6135, 0.614, 0.614, 0.6145, 0.6145, 0.615, 0.615, 0.6155, 0.616, 0.616, 0.6165, 0.6165, 0.617, 0.6175, 0.618, 0.6185, 0.6185, 0.619, 0.619, 0.6195, 0.62, 0.6205, 0.621, 0.621, 0.6215, 0.6215, 0.622, 0.6225, 0.623, 0.6235, 0.624, 0.6245, 0.625, 0.6255, 0.626, 0.626, 0.6265, 0.6265, 0.627, 0.6275, 0.6275, 0.628, 0.628, 0.6285, 0.629, 0.6295, 0.63, 0.63, 0.6305, 0.6305, 0.631, 0.6315, 0.632, 0.6325, 0.633, 0.6335, 0.634, 0.6345, 0.635, 0.6355, 0.636, 0.636, 0.6365, 0.6365, 0.637, 0.6375, 0.638, 0.6385, 0.639, 0.6395, 0.64, 0.64, 0.6405, 0.6405, 0.6405, 0.641, 0.641, 0.641, 0.641, 0.6415, 0.6415, 0.6415, 0.6415, 0.642, 0.642, 0.642, 0.6425, 0.6425, 0.643, 0.643, 0.6435, 0.6435, 0.644, 0.6445, 0.645, 0.6455, 0.646, 0.646, 0.6465, 0.6465, 0.647, 0.6475, 0.6475, 0.648, 0.648, 0.648, 0.6485, 0.6485, 0.6485, 0.649, 0.649, 0.649, 0.649, 0.6495, 0.6495, 0.6495, 0.6495, 0.65, 0.65, 0.6505, 0.651, 0.6515, 0.6515, 0.652, 0.652, 0.6525, 0.653, 0.653, 0.6535, 0.6535, 0.654, 0.6545, 0.655, 0.6555, 0.656, 0.6565, 0.6565, 0.657, 0.657, 0.657, 0.6575, 0.6575, 0.658, 0.6585, 0.659, 0.6595, 0.6595, 0.66, 0.66, 0.6605, 0.661, 0.6615, 0.662, 0.6625, 0.6625, 0.663, 0.663, 0.663, 0.6635, 0.6635, 0.6635, 0.664, 0.664, 0.6645, 0.665, 0.6655, 0.666, 0.6665, 0.6665, 0.6665, 0.667, 0.667, 0.667, 0.667, 0.6675, 0.6675, 0.6675, 0.6675, 0.668, 0.668, 0.668, 0.668, 0.668, 0.668, 0.6685, 0.6685, 0.6685, 0.6685, 0.669, 0.6695, 0.67, 0.67, 0.6705, 0.6705, 0.6705, 0.671, 0.671, 0.6715, 0.672, 0.6725, 0.673, 0.6735, 0.674, 0.6745, 0.675, 0.6755, 0.6755, 0.676, 0.676, 0.6765, 0.677, 0.6775, 0.6775, 0.678, 0.678, 0.6785, 0.679, 0.6795, 0.68, 0.6805, 0.681, 0.6815, 0.682, 0.6825, 0.683, 0.6835, 0.684, 0.6845, 0.6845, 0.685, 0.685, 0.6855, 0.6855, 0.6855, 0.686, 0.686, 0.686, 0.686, 0.686, 0.6865, 0.6865, 0.6865, 0.6865, 0.687, 0.687, 0.6875, 0.6875, 0.688, 0.688, 0.688, 0.6885, 0.6885, 0.689, 0.689, 0.6895, 0.6895, 0.69, 0.69, 0.6905, 0.6905, 0.691, 0.6915, 0.6915, 0.692, 0.692, 0.692, 0.6925, 0.6925, 0.6925, 0.6925, 0.693, 0.693, 0.693, 0.693, 0.6935, 0.6935, 0.694, 0.6945, 0.6945, 0.6945, 0.695, 0.695, 0.695, 0.695, 0.6955, 0.6955, 0.6955, 0.696, 0.696, 0.6965, 0.697, 0.6975, 0.698, 0.6985, 0.6985, 0.6985, 0.699, 0.701, 0.7015, 0.702]

Submission Instructions

Submit your work to Gradescope. There, you will need to upload a PDF file with the written answers (including the plots) to all the questions above. As well, there will be a link to upload your Python code. Make sure that the signature matches the above signature as we will check it against other types of data.