Report for Mid-term Project

First Author
Soumik Dey
zionsoumik@gmail.com

Second Author
Robert Haralick
rharalick@gc.cuny.edu

Abstract

This report is about the Mid term project assigned to us and the way we have done it. It shows how to adapt the Bayesian rules to a dataset by quantizing and then smoothing it to maximize expected gain.

1 The Tasks

There were two tasks given to us.

1.1 Task 1

- Given a measurement space $D = \times_{n=1}^{N} L_n$
- Given $d_1, d_2, d_3, ..., d_n$ compute linear addresses
- Input k number of classes
- Input $P(d|c)$ the class conditional probabilities
- Input $P(c)$ the prior probabilities
- Compute $P(c|d)$
- Compute the Discrete Bayes Rule $f_d(c)$
- Compute the K x K Confusion Matrix
- Compute the Expected Gain

1.2 Task2

- A dataset with 5 real valued attributes between 0 and 1 and 2 classes and 10,000,000 rows
- A class prior probability 0f 0.4 for class 0 and 0.6 for class 1
- Economic gain matrix

$$\begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix}$$

- A memory M of 10,000 for class conditional probabilities
- Split the data into 3 sets training,evaluation and test and through optimization find out the discrete Bayes rule and calculate the expected gain

2 Methodology

2.1 Task 1

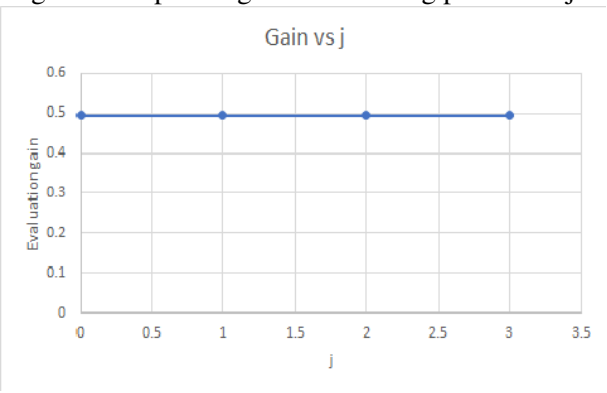
- To create the linear addresses to refer to each d
- $P(c|d)$ is calculated by $P(c|d) = \frac{P(d|c).P(c)}{\sum_{n=1}^{N} P(d|c).P(c)}$ where P(c) is prior probability
- Economic gain matrix is taken into account to calculate f_d
- We then calculate Bayes rule f_d which is determined by calculating the J value for each class $J_i = \sum_{i=1}^{K} P(c|d) * gain[i][c]$
- Then we calculate $gain = \sum_{n=1}^{N} gain[assigned][true] * P(c|d_n)$

2.2 Task 2

2.2.1 Finding the optimal number of bins

- For Task 2 the methodology needs to be fleshed out a little. I had a python file called create_sets.py which divides the dataset into 3 equal parts training, evaluation and test.
- The first step is the quantizing the real valued attributes into discrete values so that we can construct rules for it.
- We do so by doing an initial quantization of n/10 bins of the trainig set where n is the number of data points in each dimension and calculating the entropy $H_i = -\sum_{k=1}^{K} p_k \log_2 p_k - \frac{n_0 - 1}{2N \ln 2}$

Figure 1: Expected gain vs smothing parameter j



- Then I calculate $f_j = \frac{H_j}{\sum H_j}$ and then calculate the optimal number of bins $L_i = \lfloor M^{f_j} \rfloor$

2.2.2 Quantizing the final bin boundaries

- After this we start with a uniform quantization for the training set that means all bins having equal probability for each bin of dimension i according to the L_i.
- We then fix a perturbation amount which is 1/0 based on a perturbation boundary that is less than 1/4 of the difference of the two extremes.
- We then perturb each bin boundary of each dimension according to this perturbation amount and calculate the corresponding expected gain.
- We keep the setting with the largest expected gain.

2.2.3 Smoothing

- We then turn to smoothing so that we don't have any bins with very less probabilties or 0.
- M bins N observations , Smooth probabilities: Try k = jN/20M, j 1, 2, 3, 4
- Determine the smoothed probabilities using training set
- Calculate the Expected Gain using evaluation set and keep the setting with largest expected gain
- Using that setting on the test set we calculate the final expected gain.

3 Results

The Results are that I got an expected gain of 0.49540 with the j=0,1,2,3,4. The graph of k is as shown in Figure 1.