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

$$P(c|d) = \frac{P(d|c).P(c)}{\sum_{n=1}^{N} P(d|c).P(c)} \quad (1)$$

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

$$J_i = \sum_{i=1}^{K} P(c|d) * gain[i][c] \quad (2)$$

- Then calculate the confusion matrix which have $P(c^j|c^k)$ where c^j is the true class and c^k is the assigned class

$$P(c^j|c^k) = \sum_{d \in D} f_d(c^k) P(c^j, d) \quad (3)$$

- Then we calculate

$$gain = \sum_{j=1}^{K} \sum_{k=1}^{K} gain[c^j][c^k] * P(c^j|c^k) \quad (4)$$

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_sets1.py which divides the dataset into 3 equal parts training, evaluation and test randomly.
- 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 training set where n is the number of data points in each dimension and calculating the entropy

$$H_i = -\sum_{k=1}^{K} p_k \log_2 p_k - \frac{n_0 - 1}{2N \ln 2} \tag{5}$$

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

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. This is done by sorting each attribute and assigning equal count bin. For the sake of rounding all even iterations of the equal probability binning has been ceiled and odd ones floored.
- The next step find the optimal bin boundaries. This is done by randomly selecting a dimension and a bin and perturbing it's boundary and calculating $P(c|d)$s by the equation (1).
- Using those $P(c|d)$s we use the evaluation data set which will be quantized accordingly and calculate the gain for each perturbation and if we see that the economic gain is greater than the previous iteration we keep the changed bin boundaries. We do this until for 200 iterations we do not see any increase in the economic gain.
- We however go through each dimension one more time and divide each bin boundary into a flexible perturbation boundary of 1/4 each side. We then further divide this quantity into 10 steps and keep on perturbing each bin boundary for each dimension starting from the first and calculate $P(c|d)$ s.

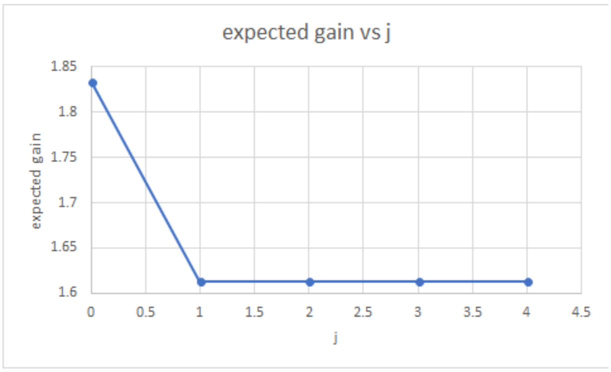

Figure 1: Expected gain vs smoothing parameter j

- If we see increase in the economic gain for these iterations we update the bin boundaries.
- 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 probabilities or 0.
- M bins N observations , Smooth probabilities: Try $k = jN/20M, j \in \{0, 1, 2, 3, 4\}$
- For smoothing we calculate k and for each k we go through each linear address and calculate for bin m the count t_m and the volume v_m and see if t_m¡k. Let $m_1, m_2, ..., m_n$ be closest n bins with hamming distance 1 such that

$$\sum_{i=1}^{n} t_{m_i} > k \tag{6}$$

and

$$\sum_{i=1}^{n-1} t_{m_i} > k \tag{7}$$

and the updated volume V_m^*

$$V_m^* = \sum_{i=1}^{n} v_{m_i} \tag{8}$$

- Determine the smoothed probabilities

$$p_m = (\alpha b_m / V_m^*) v_m \tag{9}$$

where

$$\alpha = \frac{1}{\sum_{m=1}^{n} b_m v_m / V_m^*} \tag{10}$$

- Calculate the Expected Gain using evaluation set which is quantized accordingly and keep the setting k_max with largest expected gain

- Using that bin probabilities and k_max on the test set which was quantized with same boundaries we calculate the final expected gain.

3 Results

I got the optimal bin number for all the dimensions to be 7. The Results are that I got an expected gain of 1.834 with the $j \in \{0, 1, 2, 3, 4\}$. The graph of k is as shown in Figure 1. The boundaries at the end of task 2 are in file ”boundaries6.txt”. The confusion matrix is

	0	1
0	0.4762	0.0219
1	0.025	0.4767

The accuracy is 0.953.