Harun Sasmaz

59900

Homework 3 Report

In this homework, we are asked to implement a Naïve Bayes’ Classifier. The data set is the same from the previous homework. There are 5 classes such as A, B, C, D, E. We have 25 data points for each class for training, and 14 data points for each class as test data. First part of the homework asks to separate this data set into test and training data. For the next part, we are asked to calculate prior probabilities and pcd values.

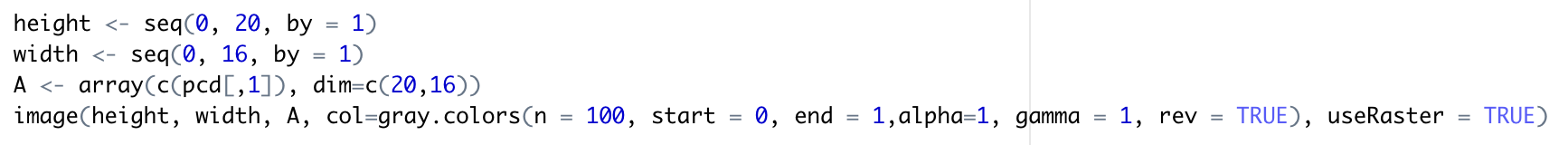
For prior probabilities, since each class has 25 data points and there are 5 classes, prior probability is 1/5 for each class. To calculate pcd values, I used colSums() function of R. Then, pcd[, i] for i = 1…5 can be calculated as column sums of class i divided by total number of data points in that class, which is 25. Briefly, for each class i, it can be computed as;

A picture containing text

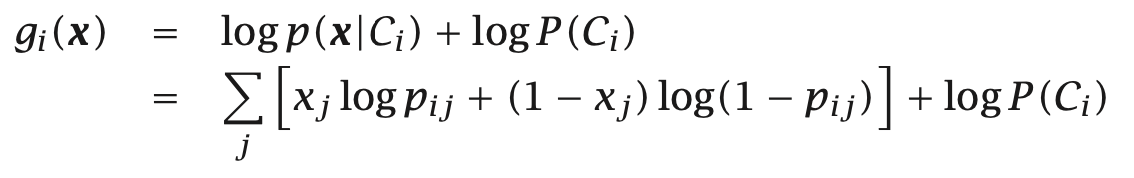
Description automatically generated

Finally, pcd has dimension 320x5.

For the next part, to draw pcd values as images, I used image () function of R. To be able to draw it as a 20x16 pixels image, I created a 2D array from pcd[, i] for each class. Given is a sample for drawing A letter. Note that, images are somehow left-rotated.



For the next part, we need to declare score functions for each class and generate the confusion matrix for train data. Score functions are defined as follows;



We can use calculated pcd values which contains densities of all classes, and we have constant prior probability for all classes as 0.2. Note that, this equation is taken from textbook “Introduction to Machine Learning 3rd ed.” by Ethem Alpaydin, chapter 5.7. We are able to use this because for a data point, each feature is independent and either 0 or 1, so it is Bernoulli.

Finally, we are able to calculate confusion matrices for train and test data. First of all, I calculated predicted labels which is an output of score functions. For each data point in the train data, I applied all score functions and pick the class which gives the maximum score. Once predicted labels are computed, I compared it with the actual labels class by class and generated the confusion matrix as usual. I also did the same thing for test data. First, computed predicted labels from test data, compared them with the actual labels and generated the confusion matrix from that comparison.