Density Estimation and Classification of Images

Abstract

The Naïve Bayes Classifier has applications in density estimation and classification, including on image classification. Using a set of examples, the machine can classify images as "0" or "1" depending on the digit that appears in an image. This method of image recognition has many practical applications in areas such as medicine and technology. We make use of a dataset consisting of 70,000 handwritten images: 60,000 training images and 10,000 in the testing images.

Implementation

The first step in implementing the Naïve Bayes Classifier is to make four separate arrays (two for the training data and two for the test data) and compute the average brightness and standard deviation of the brightness for each image in the data. Then, I computed eight parameters which are the means of averages and standard deviations, and the variances of the averages and standard deviations for each feature and digit combination. I found the gaussian for the means and standard deviations using the various parameters. Using the gaussians, I calculated the posteriors which is the product of the x_i's across all the features multiplied by 0.5 (P(y)). The classification was done by taking whichever class had the larger posterior. For example, for the posteriors computed using the data from digit 0, I produced an array of 1's and 0's by doing posterior0_digit0 > posterior0_digit1, where posterior0_digit0 used the parameters for digit 0. To find the accuracy, I summed this array and divided by the length of the same array. A similar process was used for the accuracy of digit 1.

During this project, I ran into several issues as well as important points. One of these was that I had redundant code. For example, I had written my own version of the functions for finding the average and standard deviation of a NumPy array. This was unnecessary as there are already built-in implementations for these functions. Another problem was a conceptual one. I was comparing the posteriors between two different images rather than comparing the posteriors of each image. I was supposed to compute two gaussians for each image; one with the image and digit 0 parameters, and another with the same image and digit 1 parameters. Initially, I was computing the gaussian for an image with only the same digit parameters. For example, I would use a digit 1 mean and digit 1 parameters for f1, but I would not use the digit 0 parameters for f1 for the same image.

I also had to try to make the code more concise and easier to read. For example, I made functions out of the various tasks required, such as computing the gaussian or classifying the labels. Also, I used "enumerate" in the for loops and got rid of the counting variable. Another way I made the code easier to read is that I used appropriate names for my variables. Initially had used p1, p2, …, p8 as the names of the parameters. I changed this to mean_feature1_digit0, variance_feature_digit0, … variance_feature2_digit1. This made it easier to recognize what the variable represents. An improvement I could have made in my code is making a function out of the for loops that iterate through a vector. For instance, when I iterated through the training data to compute the mean and standard deviations, I could have made a function that iterates through a vector and called that function four times (twice on the training data and twice on the test data) instead of using four for loops. This makes the code concise and less redundant.

Results

The parameter values I obtained for student id 1226806128, respectively, for the mean and variance of f1 for digit 0, the mean and variance of f2 for digit 0, the mean and variance of f1 for digit 1, and the mean and variance of f2 for digit 1 are: 44.2550160714, 114.441671736, 87.4843946561, 100.917755338, 19.3447410714, 31.067315289, 61.3173732748, and 81.6127967054, where f1 is the feature representing the average brightness of each image and f2 is the feature representing the standard deviation of the brightness of each image. The accuracy for predicting new labels for test set 0 is 0.917346938776. The accuracy for predicting new labels for test set 1 is 0.923348017621.

One fact that I noted from the results above is that the mean and variances for digit 0 are generally greater than the mean and variances for digit 1. Also, the means for feature 1 (average brightness) are smaller than the means for feature 2 (standard deviation of brightness). This could be accounted for by the differences in brightness between an image with digit 1 and digit 0. Furthermore, we would expect the standard deviations to be smaller than the means in general because of the differences in brightness. It makes sense that the accuracy of digit 1 is higher than that of digit 0 due to their differences. I noticed that there is more test data for digit 1. In all, the results I received are logical and the accuracy of this classifier is reasonable.