PROJECT ONE REPORT – MLE AND NAIVE BAYES

First, a quick summary of the tasks of this project. This project involved reading in pictures of handwritten pictures of either '1' or'0'. We gathered this information from the MNIST data set freely available online. We broke these pictures down into a large training set of pictures and a smaller testing set of pictures. Using this training set, the task at hand was to extract features from each of these images; these features included the average brightness of each pixel for a given image and the average of the variances of every row for a given image. Since we already knew what class the training sets were, we used these features to come up with the estimated values to maximize the probability of falling into each class respectively. Using these estimated values, as well as the corresponding probability distribution and it's function, we were able to use Bayes Theorem in order to calculate the probability that a new image without a label would be either a one or zero based off of the respective image's features. Since we also know the labels for the testing data, when we use the Naive Bayes Classifier on the testing image data set, we can go back and check how correct we were. After all of the data trained the learning algorithm and all of the testing data was ran through the classifier, we went back and calculated the accuracy of our classification.

The first thing that should be mentioned is how did we estimate these parameters (what formula?) and what were the results of the estimation? In order to compute the parameters, we first need to take into account of the assumptions in this project. The assumptions include that the data is a normal 2-D distribution (Gaussian). This means we are dealing with the functions and formulas for a normal distribution. Since we will be using Naive Bayes later on, which involves conditionally independent probabilities, we can break this likelihood into four 1-D normal distributions – one for each feature of each class. Thus, in order to estimate parameters for the maximum likelihood of a certain classification, we need to look at the Likelihood Function for a normal distribution:

$$L\left(\mu,\sigma^2;x_1,\ldots,x_n\right) = \left(2\pi\sigma^2\right)^{-n/2}\exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{n}(x_j-\mu)^2\right)$$

Then, in order to make computation easier, we use the log of this Likelihood Function – this is often called the Log Likelihood Function, which can be denoted as:

$$l\left(\mu,\sigma^2;x_1,\ldots,x_n\right) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln\left(\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{j=1}^{n}(x_j-\mu)^2$$

Using this Log Likelihood Function, we can find the values of mu and sigma squared such that the likelihood is maximized for those estimated values. This is done by taking the equation, and finding the supremum values for mu and sigma squared, respectfully. When we try to maximize the arguments, we get the equations of the sample mean for mu and the sample variance for sigma squared. When we actually compute these values for our training set, we can break this down into mu values for features

x1 and x2 for class zero and class one and sigma squared values for x1 and x2 for class zero and class one. These end up being:

Class Zero:
- Mu for x1 and x2 = 43 and 6240, respectively.
- Sigma squared for x1 and x2 = 115 and 1879776, respectively.

Class One:
- Mu for x1 and x2 = 18 and 2514, respectively.
- Sigma squared for x1 and x2 = 32 and 1126222, respectively.

The second thing that should come to attention regarding the distributions and the Naive Bayes Classifier is the Probability Density Function (PDF). As a note, this is a good expression to represent the distributions we have, just with the corresponding mu and sigma squared values. Since we are breaking these down into several 1-D normal distributions, the PDF for each of these can be written as:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right), \sigma > 0$$

Now, these distributions, given by the probability distribution function, of all of these normal 1-D distributions will be used in our Naive Bayes Classifier. Let us again look at the assumptions, which include: normal distributed, all samples are independent, and that the prior probability for both y=0,1 is 0.5 (p(y=0) = p(y=1) = 0.5). Now, when we take all of this into the equation, we can look at the Naive Bayes Classifier, which runs on the conditional probability for all of the features. Since we are making the assumption that our samples are conditionally independent, we can take the denominator out of the equation for Bayes Theorem. Thus, that leaves us with the probability of a certain image x containing features x1 and x2 being of class y as:

$$p(C_j \mid X) \propto p(C_j)\prod_{k=1}^{d} p(x_k \mid C_j)$$

Where Cj are the classes, so C can be 0 or 1. Then we have the prior probability on the RHS of the equation of p(Cj). If we remember our assumption in the handout, that p(C = 0) = p(C = 1) = 0.5. We can then even remove another part of the equation out and substitute that with a constant of 0.5. So all we have left now is the product of the probabilities of each variable for each class times the constant for prior probability. This leaves us with just these equations for the Naive Bayes Probabilities:

$$\begin{aligned} p(X|y=0) &= p(y=0)p(x1|y=0)p(x2|y=0) \\ &= 0.5 * p(x1|y=0)p(x2|y=0) \\ p(X|y=1) &= p(y=1)p(x1|y=1)p(x2|y=1) \\ &= 0.5 * p(x1|y=1)p(x2|y=1) \end{aligned}$$

This is really easy to solve at this point, because we can find p(xi|y={0,1}) for any new image x with features x1 and x2 with the PDF function we defined earlier using mu and sigma squared data from the training set we ran through our leaning algorithm.

With all of this implemented, when running the testing data through the classifier I wrote, the results were a 94.18 percent accuracy with the testing set of zero, a 93.13 percent accuracy with the testing set of one, and lastly the overall testing accuracy is 93.62 percent.

Lastly, I want to give some personalized details of the code I wrote for the project. Ran by itself, it prints out the results of the three values of accuracy before mentions; however, I added in a little tool to help grade it and test it. If ran with the arguments --testing 1 then it marks the testing flag throughout the project to 1 and thus prints out more information for testing purposes instead of just ran normally. Thus, runnnig ‘python main.py’ will print out the three accuracy values, but when running ‘python --testing 1’, it will print out information including: the feature vectors of all four data sets, 2-D plots of the feature vectors, estimated parameter values, the labels given for both sets of classes after the testing data was ran through, and the final accuracy numbers.