DENSITY ESTIMATION AND CLASSIFICATION

Name- Kunal Kumar
ASU ID-1217167445

Introduction:

In this project we are given MNIST dataset which contains 70000 images of handwritten digits, 60,000 training images and 10,000 testing images. We have to use the digits 7 and 8 in the images to perform parameter estimation and classification. The test and training dataset are represented by "trX" and "tsX" respectively. Their corresponding labels or targets are stored in "trY" and "tsY" respectively.

Given dataset details:
1. Number of samples in a training set: "7":6265; "8":5851.
2. Number of samples in testing set: "7":1028, "8":974

Feature Extraction:

The data is extracted from "mnist_data.mat" file. In the training and testing data, each row represents a digit where as in the Label data, 0 and 1 represents 7 and 8 respectively. The testing and training dataset have 784 features – which represent the pixel values of each image. We need to extract only two features i.e. mean and standard deviation for each image. Hence, we need to extract mean and standard deviation for 12116 images of the training dataset and 2002 images of the training dataset.

Image Class	Mean	Standard Deviation
7	0.11452	0.28755
8	0.15015	0.32047

Naïve Bayes Classification:

Naive Bayes Classifier is a probabilistic classifier based on Bayes theorem where the features of the class are independent of each other. The probability that an image belongs to a particular Class is represented as -

P (Y=Class / X) = P (X/Y=Class) * P (Y= Class) / P(X)

Where P (Y=Class/X) is known as the **posterior**,
P (X/Y=Class) is known as the **likelihood**,
P (Y= Class) is known as **Class Prior Probability**
P(X) is known as **Predictor Prior Probability.**

But, the condition for the Naïve Bayes is that all the features are independent. We have 2 features i.e. Mean and Standard Deviation. So, we can write the equation as

P (Y=Class/X) = P (Class/X1) * P (Class/X2)

where X1 is **Mean**
X2 is **Standard Deviation**

Since, it follows Normal distribution, we have made a function p_x_given_y as

```
def p_x_given_y(x, mean, variance):
    p_x_give_y = (1 / (np.sqrt(2 * np.pi * variance)))
                  * np.exp(-(x - mean) ** 2 / (2 * variance))
    return p_x_give_y
```

Finally, we can write the posterior probability of class 7 and class 8 as:

```
post_prob_class = prior_prob_class \
              * p_x_given_y(testing_set_mean, mean_class.mean(),
                mean_class.var()) \
              * p_x_given_y(testing_set_sd, sd_class.mean(),
                sd_class.var())
```

We calculate the numerator of posterior probability for class 7 and class 8 and compare the values.
If the numerator of posterior probability of class 7 is greater than the numerator of posterior probability of class 8, then the image is classified as belonging to Class -7 or vice versa.

```
value_compare = np.greater(post_prob_8, post_prob_7)
```

Finally, we calculate the accuracy of the individual image 7 and image 8.
And at last we've calculated the accuracy of the Naïve Bayes Classification.

Logistic Regression Classification:

In Logistic Regression, for a given set of inputs, we try them to assign them to 0 or 1. LR predicts the probability to assign the inputs in one of the two categories. We will be using the sigmoid function to determine the probability. The function maps any real value into another value between 0 (representing Class -7) and 1 (representing Class – 8).

The hypothesis is represented as

$$h_\theta(\mathbf{x}) = g(\theta^\top \mathbf{x})$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

The parameter of the logistic function is θ, which is the coefficient or the weight. We need to calculate the parameter θ, such that the log-likelihood is maximized –.

w∗ = argmax**w**l(θ)

w∗ = argmaxθ ∑ [$y(i)\theta tx(i)$ − log❪1 + exp(θ$tx(i)$)❫] n i=1

We cannot really solve for w* analytically (no closed-form solution). We will use Gradient Ascent in order to maximize the equation.

The gradient ascent equation which is used to find the θ is given by –

θ(k+1) = θ(k) + η∇θ(k)l(θ)

η > 0 is called the learning rate,

l(θ) represents the log likelihood function

gradient = np.dot(X.T, (h - y)) / y.shape[0] theta = theta - learningRate * gradient

where, Learning Rate = 0.001

```
thetas = log_reg(train_feature, train_label, num_iterations=150000,
         learning_rate=0.001,
         add_intercept=True)
```

We will substitute the above equation in the gradient ascent equation. We set the number of iterations as 150000 and the learning rate as 0.001.

If the result of the sigmoid function is more than 0.5, then it is classified as belonging to Class-8., else Class – 7.

The value of the θ parameters determined by Logistic Regression are-

[23.0058898 246.35633793 -180.19141079]

Results:

Classifier	Class 7	Class 8	Overall Accuracy
Naïve Bayes	75.97	62.73	69.53
Logistic Regression	78.59	85.01	81.71

The computation time for fitting the Naïve Bayes classifier is substantially less as compared to Logistic Regression Classifier.
It must be noted that Naïve Bayes is a very simple classifier. Even though we know that Mean and SD as feature are correlated, we consider them to be independent for the sake of simplicity.

The Accuracy of Logistic Regression is much better compared to Naïve-Bayes Classifier