DENSITY ESTIMATION AND CLASSIFICATION

Francis Mendoza

ASUID: 1213055998

CSE575- Statistical Machine Learning

fmendoz7@asu.edu

1. **INTRODUCTION**
    a. I extracted a subset of the MNIST dataset, particularly dealing with the digits "7" and "8". In this project, I used MLE Density Estimation, Naïve Bayes classification and Logistic Regression to find the average of all pixel values in the image, as well as the standard deviation of all pixel values in the image. The entire file was first developed on the Jupyter Notebook environment and then converted into a .py file using the following command
        i. `jupyter nbconvert --to script Mendoza-DRAFTFORPY_Project1.ipynb`
2. **FEATURE EXTRACTION**
    a. The extracted dataset contains training and testing samples for "7" and "8", stored as a dictionary.
        i. The key is a string literal, denoting training or testing and which digit
        ii. The value is a list of lists, denoting the brightness output of pixels
    b. Training Samples:
        i. "7": 6265 rows
        ii. "8": 5851 rows
    c. Test Samples:
        i. "7": 1028
        ii. "8": 974
3. **NAÏVE BAYES CLASSIFICATION**
    a. The results from the initial feature extraction (calculating the means of both the means and the standard deviations) follows as such:

    TRAINING SET

i. $trX\ MEAN\ OF\ MEANS$ 8: 0.1501559818936975

ii. $trX\ MEAN\ OF\ STDev$ 8: 0.3206804173181901

iii. $[0.1501559818936975, 0.3206804173181901]$

iv. $Covariance\ trX, 7 = \ 0.0011547856877473894$

v. $Covariance\ trX, 8 = \ 0.0015146160624154846$

- Since the covariance is small, this implies that there is no strong correlation between mean and standard deviation (and thus, are strongly independent variables)

TESTING SET

i. $tsX\ MEAN\ OF\ MEANS$ 7: 0.11452769775108766

ii. $tsX\ MEAN\ OF\ STDev$ 7: 0.28774013146823724

b. Due to the results of the Naïve Bayes algorithm, I was able to find that the training set consistently displayed more values learning towards “7” then “8”, as it had a higher probability value for the former (0.055463849455265765) than for the latter (0.00924397490921096)

c. Unfortunately, I was unable to acquire the respective accuracy of the training data compared to the test dataset. It appears I need additional practice with Python for future assignments and will be visiting office hours regularly.

```
--------------------
Y Value: 0.0
SEVEN is: 1.2808797392307636
EIGHT is: 1.1148796955931017
IT IS 7
--------------------
Y Value: 0.0
SEVEN is: 1.2808797392307636
EIGHT is: 1.1148796955931017
IT IS 7
--------------------
Y Value: 0.0
SEVEN is: 1.2808797392307636
EIGHT is: 1.1148796955931017
IT IS 7
--------------------
Y Value: 0.0
SEVEN is: 1.2808797392307636
EIGHT is: 1.1148796955931017
IT IS 7
--------------------
```

In [32]:

```python
length = len(actualVector)
print("SEVEN PROBABILITY VALUE: " + str(aCounter/length))
print("EIGHT PROBABILITY VALUE: " + str(bCounter/length))

#Hence, because there are greater instances of seven to eight, it belongs at 7
```

```
SEVEN PROBABILITY: 0.055463849455265765
EIGHT PROBABILITY: 0.00924397490921096
```

4. LOGISTIC REGRESSION

a. Although the logistic regression code was implemented, due to the complexity and my current lack of understanding on advanced data manipulation into numpy, I was unable to find the decision boundary using gradient descent. Because of this, this code block was commented out to get the rest of the file to compile properly.

5. RESULTS

a. Due to the incomplete gathering of data, the lack of proper accuracy values gathered and the lack of a functioning logistic regression, the results could not be conclusively determined. I would like to come in during office hours so I will do better in the future.