ASU Arizona State University

CSE 575: Statistical Machine Learning
Project Part 1 – Density Estimation and Classification

Submitted by:
Abhishek Mohabe
1219468432

Naïve Bayes Classifier

The formula used for the calculation of the average and standard deviation is as follows:

```
trainingAvg.append(sum(trainingFiledata[i])/trainingFiledataLen)
trainingStddev.append(statistics.stdev(trainingFiledata[i]))
```

The average and standard deviation of each image is stored in a list:

For training set:

```
trainingAvg
trainingStddev
```

For testing set:

```
testingAvg
testingStddev
```

Estimated values for the defined parameters are as follows:

```
Abhisheks-MacBook-Pro:Project 1 abhishek$ python3 naivebayes.py
Values for X1
Average-class 0:  0.11452769775108732
Average-class 1:  0.15015598189369694
Standard Deviation-class 0:  0.030634849714227558
Standard Deviation-class 1:  0.038635790154946476
Values for X2
Average-class 0:  0.28774013146823685
Average-class 1:  0.32068041731819086
Standard Deviation-class 0:  0.03822852104266739
Standard Deviation-class 1:  0.03998900094264787
```

Scipy library was used to calculate the 2-d normal gaussian distribution. It has been calculated as follows:

```
gaussian0X1 = scipy.stats.norm(avg0X1, std0X1)
gaussian1X1 = scipy.stats.norm(avg1X1, std1X1)
gaussian0X2 = scipy.stats.norm(avg0X2, std0X2)
gaussian1X2 = scipy.stats.norm(avg1X2, std1X2)
```

The accuracy for the naïve bayes classifier is **52%**

Logistic Regression Classifier

The formula used for the calculation of the average and standard deviation is as follows:

```
trainingAvg.append(sum(trainingFiledata[i])/trainingFiledataLen)
trainingStddev.append(statistics.stdev(trainingFiledata[i]))
```

The average and standard deviation of each image is stored in a list:

For training set:

```
trainingAvg
trainingStddev
```

For testing set:

```
testingAvg
testingStddev
```

Expression for the sigmoid function and the label prediction is defined as follows:

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def predict(X):
    linearModel = np.dot(X, weights) + bias
    yPredicted = sigmoid(linearModel)

    yPredicted_class = [1 if i > 0.5 else 0 for i in yPredicted]
    return yPredicted_class
```

Linear model is being calculated using the sense of w0 + w1.x1 + w2.x2 which is then passed to the sigmoid function for 1000 iterations of weight calculation.

```
for _ in range(epochIters):
    linearModel = np.dot(X, weights) + bias
    yPredicted = sigmoid(linearModel)
    dw = (1 / nSamples) * np.dot(X.T, (yPredicted - y))
    db = (1 / nSamples) * np.sum(yPredicted - y)

    weights -= learningRate * dw
    bias -= learningRate * db
```

Then the accuracy is calculated as shown below:

```
def accuracy(yTrue, yPred):
  accuracy = np.sum(yTrue == yPred) / len(yTrue)
  return accuracy
```

The accuracy using the Logistic Regression classifier is **72%.**