Report

Project Part 1: Density Estimation and Classification using Fashion-MNIST:

Submitted by,
Shashank Davalgi
1219510734
sdavalgi@asu.edu

Naïve Bayes Classifier :

Formula used to estimate the parameters:

The features used for the classification are the **Average** and **Standard Deviation.**

The average and standard deviation are calculated for each image and stored in a list as shown below:

```
average = []
standard_deviation = []
```

The formula used to calculate both Average and Standard deviation is shown below:

```
average.append(sum(training_data_array[i])/len(training_data_array[0]))
standard_deviation.append(statistics.stdev(training_data_array[i]))
```

Estimated values for the parameters:

The mean and standard deviation is calculated for X1 and X2 for each class and the values are as below:

```
Mean of X1 for class 0:  0.3256077664399094
Standard Deviation of X1 for Class 0:  0.11338436369948433
Mean of X1 for class 1:  0.22290531462584984
Standard Deviation of X1 for Class 1:  0.056955755259147804
Mean of X2 for class 0:  0.32024038721169107
Standard Deviation of X2 for Class 0:  0.0880463127795082
Mean of X2 for class 1:  0.3341548890116032
Standard Deviation of X2 for Class 1:  0.05707345032584217
```

Expression for the estimated normal distribution:

The 2-D normal (gaussian) distribution is estimated by using the scipy library. It is found for both the parameters of each class.

Calculated as shown below:

```
# Gaussian Distribution for x1 and x2 for both the classes 0 and 1:

gaussian_dist_x1_class_0 = scipy.stats.norm(mean_x1_class_0, std_x1_class_0)
gaussian_dist_x1_class_1 = scipy.stats.norm(mean_x1_class_1, std_x1_class_1)
gaussian_dist_x2_class_0 = scipy.stats.norm(mean_x2_class_0, std_x2_class_0)
gaussian_dist_x2_class_1 = scipy.stats.norm(mean_x2_class_1, std_x2_class_1)
```

Classification explanation by using the distribution:

Mean and standard deviation is calculated for both x1 and x2 for each class.
The gaussian distribution is calculated for each input parameters for both the classes where the inputs are the mean and standard deviation obtained from the above step.
Then, the PDF is calculated for each input variable by using **.pdf()** function of the gaussian distribution.
prob_class_0_x1_x2 = gaussian_dist_x1_class_0.pdf(testing_list_x1_class_0[i]) * gaussian_dist_x2_class_0.pdf(testing_list_x2_class_0[i]) * prob_class_0
Then, the class is predicted based on the PDF value for each input variable.

Classification Accuracy:

The accuracy for the naïve bayes classifier using normal distribution is **83.15%**

```
Accuracy for the naive bayes algorithm:  83.15 %
```

Logistic Regression Classifier:

Formula used to estimate the parameters:

The features used for the classification are the **Average** and **Standard Deviation.**

The average and standard deviation are calculated for each image and stored in a list as shown below:

```
average = []
standard_deviation = []
```

The formula used to calculate both Average and Standard deviation is shown below:

```
average.append(sum(training_data_array[i])/len(training_data_array[0]))
standard_deviation.append(statistics.stdev(training_data_array[i]))
```

Expression for the Sigmoid Function:

The sigmoid function is calculated for each class and the label is predicted as shown below:

```
def predict(X):
    linear_model = np.dot(X, weights) + bias
    y_predicted = sigmoid(linear_model)
```

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

Classification explanation by using the distribution:

Linear model is calculated by using **w0 + w1.x1 + w2.x2**
This is passed to the sigmoid function and the weights are calculated for 1000 iterations as shown below:

```
for _ in range(epoch_iters):
    linear_model = np.dot(X, weights) + bias
    y_predicted = sigmoid(linear_model)

    dw = (1 / n_samples) * np.dot(X.T, (y_predicted - y))
    db = (1 / n_samples) * np.sum(y_predicted - y)

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

Once the weights are found, predict the y values.

Classification Accuracy:

The accuracy for the naïve bayes classifier using normal distribution is **90.45%** and it is calculated as shown below:

```
accuracy = np.sum(y_true == y_pred) / len(y_true)
return accuracy
```