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

Due Feb 22 by 11:59pm **Points** 10 **Submitting** a file upload
Available until Feb 23 at 3am

This assignment was locked Feb 23 at 3am.

Project Overview:

In this part, you need to first perform parameter estimation for a given dataset (a subset from the Fashion-MNIST dataset). The Fashion-MNIST dataset contains 70,000 images of article images, divided into 60,000 training images and 10,000 testing images. We use only images for “Tshirt/Top” class and “Trouser” class in this project.

We have the following statistics for the given subset:

- Number of samples in the training set: "Tshirt": 6000 ; "Trouser": 6000.
- Number of samples in the testing set: "Tshirt": 1000; "Trouser": 1000.

You are required to extract the following two features for each image:

1. The average of all pixel values in the image
2. The standard deviation of all pixel values in the image

We assume that these two features are independent, and that each image (represented by a 2-D features vector) is drawn from a 2-D normal distribution.

To ease your effort, we have extracted the necessary images and stored them in “fashion_mnist.mat” file, which can be downloaded **here**. A description of the file can be downloaded **here**.

You may use the following piece of code to read the dataset:

```
import scipy.io

data = scipy.io.loadmat('fashion_mnist.mat')
```

The specific algorithmic tasks you need to perform for this part of the project include:

1. Extracting the features and then estimating the parameters for the 2-D normal distribution for each class, using the training data. Note: You will have two distributions, one for each class.
2. Use the estimated distributions for doing Naïve Bayes classification on the testing data. Report the classification accuracy for both the classes in the testing set.
3. Use the training data to train a Logistic Regression model using gradient ascent. Report the classification accuracy for both classes in the testing set. **Note that you are not allowed to use package like sklearn to compute the boundary. You need to implement your own version for using gradient ascent to find the solution.**

Algorithms:

MLE Density Estimation, Naïve Bayes classification, Logistic regression

Workspace:

Any Python programming environment.

Software:

Python environment.

Language(s):

Python. (MATLAB is equally fine, if you have access to it.)

Required Tasks:

1. Write code to extract features for both training set and testing set.
2. Write code to implement the Naive Bayes Classifier and use it produce a predicted label for each testing sample.
3. Write code to implement the Logistic Regression and use it produce a predicted label for each testing sample.
4. Write code to compute the classification accuracy, for both the Naive Bayes Classifier and Logistic Regression.
5. Write a short report summarizing the results, including the final classification accuracy.

Note that you are not allowed to use package like sklearn to compute the boundary.

Deliverables and due date(s):

The code and report are due by **Wednesday 17 at 11:59 PM MST**.

What to Submit:

Code:

- Acceptable file types are .py/.m or .zip.
- If you have only one file, name the file to be main.py or main.m (matlab users), and submit it.
- If you have multiple code files, please name the main file as main.py and name other files properly based on its content; Similarly, for matlab users, you should have only one main.m and other relevant .m files. Next, zip all the files and submit Code.zip file.
- Documentation comment is important and required. Be sure to read through the directions carefully to ensure you have included all necessary parts in your code.

Report:

- Acceptable File types: .pdf
- Length: 2-5 A4 pages
- Content: Include the formula that you used to estimate the parameters, the estimated values for the parameters, the expression for the estimated normal distributions, an explanation for how the distributions are used in classifying a testing sample, and the final classification accuracy for both the classes for the testing set for both the algorithms.