Project (Part 1):

PCA, Density Estimation, and Bayesian Classification

Introduction

In this project, we aim to perform Principal Component Analysis (PCA), Density Estimation and Bayesian Classification techniques on a dataset.

The project uses a subset of images (with modifications) from the Fashion-MNIST dataset. The original Fashion-MNIST dataset contains 70,000 images of objects, divided into 60,000 training images and 10,000 testing images. The dataset we use has two classes: "T-shirt" (or class 0 as we decide to call it) and class "Sneaker" (or class 1). The data is stored in ".mat" files. In the original .mat file, each image is stored as a 28x28 array.

We aim to extract this data, make it suitable to work on (using PCA), perform analysis on it (estimate density of classes) and predict our output (using Bayes classification and minimum error rate calculation) based on our study of the data.

Method

A series of steps were followed in order for us to achieve what we needed. We will be going through them in a step-by-step approach.

Step 1: Loading the data from the MATLAB files

In this step, we first loaded the data into our python environment and stored it in the form of a list. We did this to both the training and testing datasets.

Our focus is on the two types of keys in the data: x and y. The data within the key 'x' are our images, with each image being a 28x28 array. The data within our key 'y' is the feature label for each image (0 for 'Tshirt' and 1 for 'Sneaker').

We split the data into testing and training datasets for both x and y. Thus, we have X_train, X_test, y_train and y_test as our data within the python environment. We will work on this data throughout our project.

Step 2: Normalizing the data

We have to normalize the data that we have extracted from the matlab files. Normalization helps in equal scaling of features so that larger values cannot dominate the principal components, thereby ensuring equal contribution of each feature to the PCA. It also centers the data around 0, so that consistency is maintained in the calculation of covariance matrix and PCA captures directions of maximum variance with proper accuracy.

In order to achieve normalized data, we first concatenate the columns in X_train and X_test into a 784 dimension dataset so that we can apply PCA properly to it afterwards. We then use the **numpy** library to calculate the mean and standard deviation of the data and apply the formula to calculate the normalized version of each data point.
Formula: normalized_data = (data - mean)/S.D

Step 3: Principal Component Analysis
We now perform PCA on our normalized data that we have calculated in step 2. PCA (Principal Component Analysis) is used to reduce the dimensionality of data while preserving as much variance as possible. The more the variance, the more information we get from the data. PCA projects the data onto a smaller set of new axes (principal components) and simplifies the dataset, thereby making it easier to visualize, analyze, and use in machine learning models. It helps in removing noise, reducing computational costs, and improving model performance by focusing on the most informative features.
Here, we perform PCA by first calculating the covariance matrix of the training data. The formula for calculating the direction in which we get maximum variance is given by the eigenvalue of the covariance matrix. Thus, after we calculate the covariance matrix, we use that to calculate the eigenvectors and the corresponding eigenvalues to get the principal components.
Thus we get 784 principal components upon calculating the eigenvectors and eigenvalues in this step.

Step 4: Dimensionality reduction
We now try to reduce the dimensionality of the data in order for the ML model to process it easily. We take the first two principal components that we got from step 3 because it is these two components that give us the maximum information (show maximum variance) out of all the other components.
After taking the components, we project the training and testing data onto these vectors and try to visualize them.
In the scatterplot visualization, we get to see how varied the data is in both the 1st and 2nd principal components. In the figure below, we see the plot of the training and testing samples.

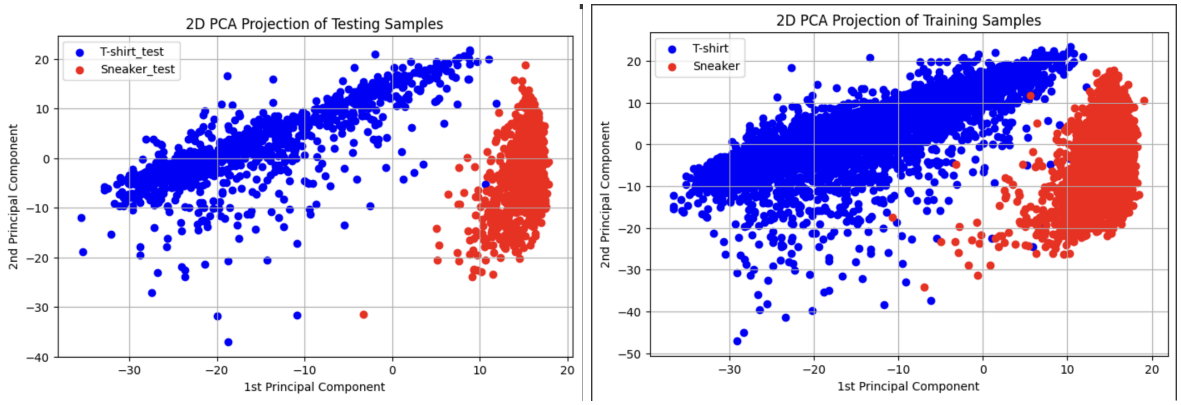

Fig 1 Scatter plots showing the projection of training and testing samples

The blue dots represent "T-shirt" samples, and red dots represent "Sneaker" samples. The data points are mostly separated, with "T-shirts" clustering on the left and "Sneakers" on the right, indicating that the two classes are distinguishable in this 2D principal component space. This separation suggests that PCA has effectively reduced the dimensionality while preserving the class separability. We also see more variance horizontally (i.e along the 1st principal component) than vertically (2nd principal component).

In the histogram visualization, we see the plots being normally distributed. This is important for the steps that follow, as we need to assume that our data is normally distributed for applying the formula for Bayes Classification and minimum error rate calculation.

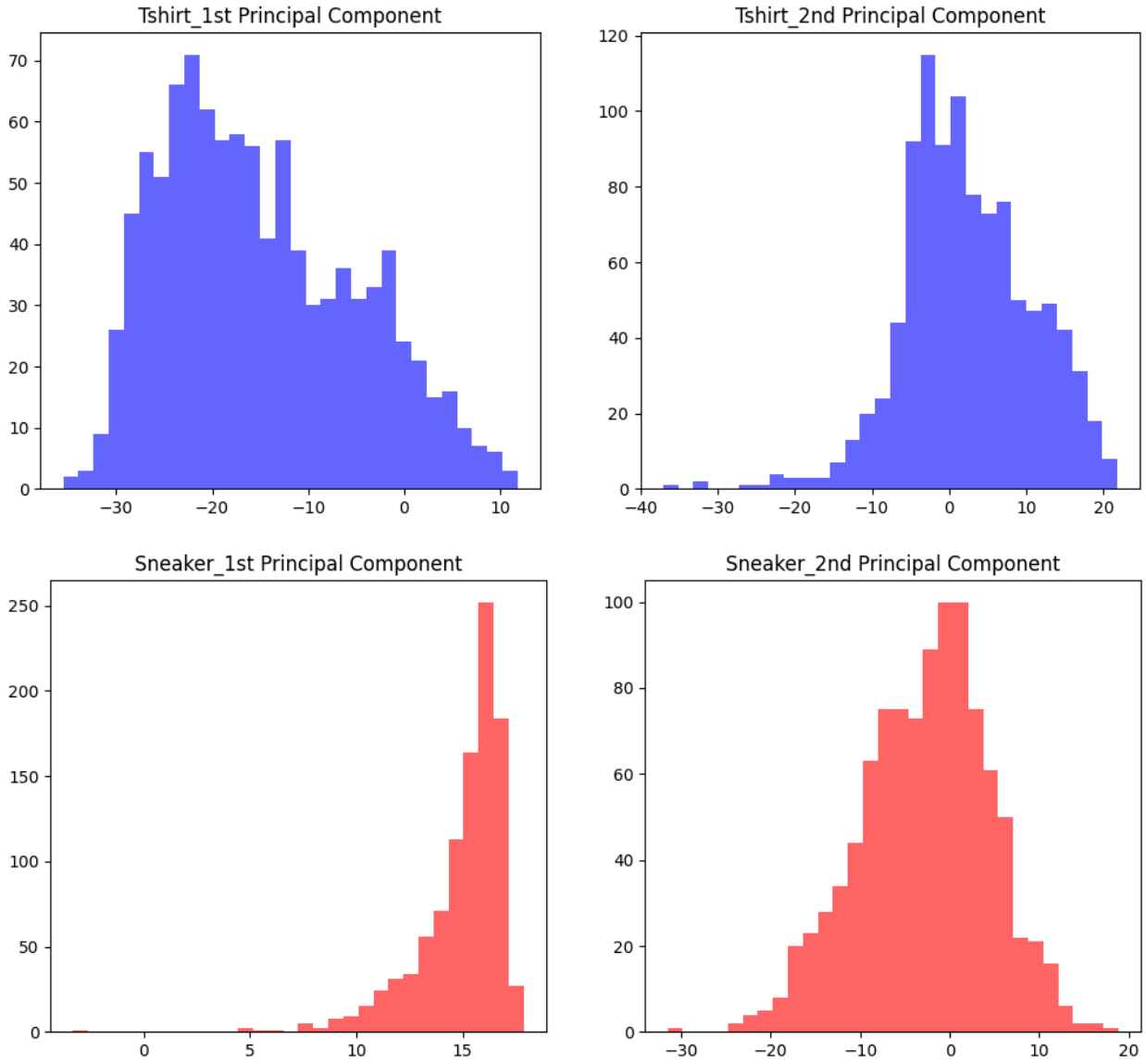

Fig 2 Histogram plots showing the distribution of the samples. We can see that the data seems to have a certain degree of skewness to it and it is not 100% normally distributed.

Step 5: Parameter Estimation
In this step, we estimate the parameters of the data, namely the mean and the variance. We do this separately for both the classes ("TShirt" and "Sneakers").
Using the numpy library, we calculate the means and the covariance matrices (i.e variances) of both the classes.

Step 6: Minimum Error Rate Classification
In this step, we aim to classify the data based on our likelihood estimate.
We use a Gaussian probability likelihood function, to estimate the likelihood based on the parameters that we estimated in the previous step.

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

Fig 3: Formula for calculating the likelihood based on mean and covariance (image taken from slide "**03-Bayesian-Decision-Theory.pdf"** prepared by instructor)

After calculating the likelihood, we use our result to classify the training data. Classification involves the following formula: **"**Decide **w1** if **P(w1).P(x|w1) > P(w2).P(x|w2)",** where **w1** and **w2** are our classes and **P( )** represents the probability. Since here we have assumed our priors (i.e P(w1) and P(w2)) to be equal we only need to compare our likelihood estimates (i.e P(x|w1) and P(x|w2)). Thus, we classify our sample into "Tshirt" if the likelihood of Tshirt is greater than the likelihood of "Sneakers".

We put our training data through this process and predict our outputs. These outputs are then compared to our given class dataset (i.e y) and our accuracy is calculated. We repeat this process for the testing data.

Results And Observations

The results that we obtained are listed as follows:

1. **PCA visualizations** (please refer to Figures 1 and 2)
2. **Parameter Estimations**
   Mean for T-shirt class: [-14.96013579 2.53759713]
   Mean for Sneaker class: [14.96013579 -2.53759713]
   Covariance matrix for T-shirt class:
   [[103.38209995 69.15661464]
   [69.15661464 73.43418283]]
   Covariance matrix for Sneaker class:

[[4.94656978 6.78163691]
[6.78163691 54.67473161]]

3. **Minimum Error Rate Classification Accuracy.**
   Training Accuracy: 99.750000%
   Testing Accuracy: 99.900000%

Based on the results, we observe that the model that uses minimum error rate classification is highly accurate and thus reliable in predicting the classes for the samples.
In **Figure 2,** we observe that the two classes seem to be properly separated and form fairly distinct clusters.
We have assumed two major things while doing this experiment:

1. Normal distribution of the samples.
2. Priors of both the classes are equal.

These two assumptions made our process easier and less complex.

Conclusion

- We were able to successfully reduce the dimension of the data while preserving most of the variance. This allowed the classifier to distinguish between the two classes with not much information loss.
- The training and testing accuracy are similar. This means that there is very little overfitting and that the model is able to generalize to new samples.
- We have received a high accuracy for the predictions. This means that the two classes (“Tshirt" and “Sneakers”) are properly separated in the PCA and they form distinct clusters.

Overall, this project demonstrates the power of PCA for dimensionality reduction and Bayesian classification, achieving both simplicity and high classification performance.