Question 1

Objective:

The code aims to perform Quadratic Discriminant Analysis (QDA) on the MNIST dataset to classify handwritten digits.

Libraries Used:

- numpy: For numerical computing.
- matplotlib.pyplot: For plotting images.
- np.load(): To load data from a .npz file.
- np.mean(): To calculate mean.
- np.cov(): To calculate covariance matrix.
- np.linalg.det(): To compute the determinant of a matrix.
- np.linalg.pinv(): To compute the Moore-Penrose pseudo-inverse of a matrix.

Data Loading:

- The MNIST dataset is loaded from the specified file path.
- Training and testing data along with their labels are extracted.

Data Preprocessing:

- Reshaping the training and testing data to flatten the images from 2D to 1D.

Functions:

c_mu_sigma(data, labels):

- Computes the mean and covariance matrix for each class in the dataset.
- Parameters:
  - data: Flattened training data.
  - labels: Corresponding labels for the training data.
- Returns:
  - mu: List of mean vectors for each class.
  - sigma: List of covariance matrices for each class.

qda(x, mu_total, sigma_total, priors):

- Performs Quadratic Discriminant Analysis (QDA) to predict class labels for the input data.
- Parameters:
  - x: Input data.
  - mu_total: List of mean vectors for each class.
  - sigma_total: List of covariance matrices for each class.
  - priors: Prior probabilities of each class.
- Returns:
  - Predicted class labels.

Model Evaluation:

- Accuracy is calculated for the test set using the QDA predictions.
- Class-wise accuracy is also calculated.

Visualization:

- Visualizes sample images from each class using matplotlib.

Outputs:

- Prints the overall accuracy for the test set.
- Prints class-wise accuracy for each digit class.

Notes:

- The code assumes that the MNIST dataset is stored in a .npz file format.
- It performs QDA which assumes each class has its own covariance matrix.
- The dataset is assumed to have 10 classes (digits 0 to 9).

Below is a detailed documentation for the provided Python code:

Objective:

The code aims to perform Quadratic Discriminant Analysis (QDA) on the MNIST dataset to classify handwritten digits.

Libraries Used:

- numpy: For numerical computing.
- matplotlib.pyplot: For plotting images.
- np.load(): To load data from a .npz file.
- np.mean(): To calculate mean.
- np.cov(): To calculate covariance matrix.
- np.linalg.det(): To compute the determinant of a matrix.
- np.linalg.pinv(): To compute the Moore-Penrose pseudo-inverse of a matrix.

Data Loading:

- The MNIST dataset is loaded from the specified file path.
- Training and testing data along with their labels are extracted.

Data Preprocessing:

- Reshaping the training and testing data to flatten the images from 2D to 1D.

Functions:

c_mu_sigma(data, labels):

- Computes the mean and covariance matrix for each class in the dataset.
- Parameters:

        - data: Flattened training data.
        - labels: Corresponding labels for the training data.
    - Returns:
        - mu: List of mean vectors for each class.
        - sigma: List of covariance matrices for each class.

qda(x, mu_total, sigma_total, priors):

- Performs Quadratic Discriminant Analysis (QDA) to predict class labels for the input data.
- Parameters:
    - x: Input data.
    - mu_total: List of mean vectors for each class.
    - sigma_total: List of covariance matrices for each class.
    - priors: Prior probabilities of each class.
- Returns:
    - Predicted class labels.

Model Evaluation:

- Accuracy is calculated for the test set using the QDA predictions.
- Class-wise accuracy is also calculated.

Visualization:

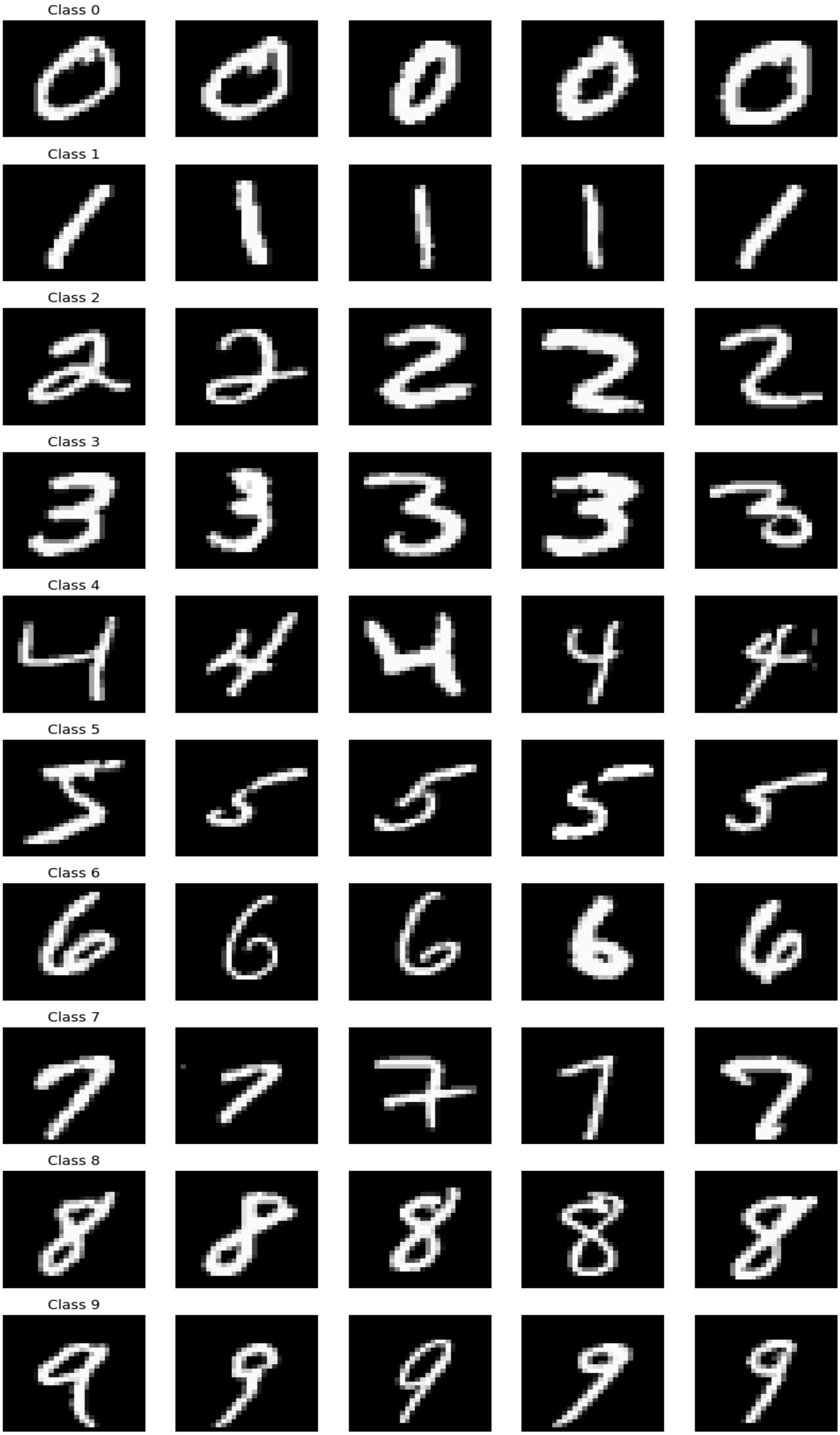
- Visualizes sample images from each class using matplotlib.

Outputs:

- Prints the overall accuracy for the test set.
- Prints class-wise accuracy for each digit class.

Notes:

- The code assumes that the MNIST dataset is stored in a .npz file format.
- It performs QDA which assumes each class has its own covariance matrix.
- The dataset is assumed to have 10 classes (digits 0 to 9).


Class 0
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class 7
Class 8
Class 9

```
Accuracy for the test set: 85.72 %
Class-wise Accuracy for Test Set:
Class 0: 93.46938775510203 %
Class 1: 67.40088105726872 %
Class 2: 93.6046511627907 %
Class 3: 87.52475247524752 %
Class 4: 90.93686354378818 %
Class 5: 79.93273542600897 %
Class 6: 89.03966597077245 %
Class 7: 86.38132295719845 %
Class 8: 88.80903490759754 %
Class 9: 82.1605550049554 %
```

Question 2

Objective:

The code aims to perform dimensionality reduction using Principal Component Analysis (PCA) and evaluate Quadratic Discriminant Analysis (QDA) classification performance on the reduced feature space using the MNIST dataset.

Libraries Used:

- numpy: For numerical computations.
- matplotlib.pyplot: For visualization.

Data Loading and Preprocessing:

- MNIST dataset is loaded from the specified file path.
- Training and testing data along with their labels are extracted and reshaped.
- Necessary functions and classes are defined.

Dimensionality Reduction using PCA:

- PCA is applied to the training data to reduce its dimensionality.
- Mean centering and covariance matrix calculation are performed.
- Eigenvalues and eigenvectors are computed.

- The top principal components (eigenvectors) are selected based on the explained variance.

Visualisation of Reconstructed Images:

- Reconstructed images are plotted using different numbers of principal components (p) for visualisation and comparison.

Quadratic Discriminant Analysis (QDA):

- A custom QuadraticDiscriminantAnalysis class is defined to perform QDA.
- It includes methods for fitting the model and making predictions.
- The model is trained using the reduced feature space obtained from PCA.

Model Evaluation:

- The QDA model is evaluated on the test set using different numbers of principal components (p).
- Overall accuracy and class-wise accuracy are calculated and printed for each p.

Functions and Classes:

plot_img(p, U, X_central, meanvals, num_images=5):

- Plots reconstructed images for visualization.
- Parameters:
    - p: Number of principal components.
    - U: Principal component matrix.
    - X_central: Mean-centered data.
    - meanvals: Mean values per pixel.
    - num_images: Number of images to plot.

QuadraticDiscriminantAnalysis Class:

- Custom class for Quadratic Discriminant Analysis.
- Methods include fit() for training and predict() for making predictions.

Output:

- Overall accuracy and class-wise accuracy are printed for each number of principal components (p).

A few samples of images plot on p=[5,10,20] and 2nd line p=784

```
Overall Accuracy for p=5: 61.52%
Class 0 Accuracy for p=5: 92.96%
Class 1 Accuracy for p=5: 10.04%
Class 2 Accuracy for p=5: 83.04%
Class 3 Accuracy for p=5: 67.72%
Class 4 Accuracy for p=5: 47.45%
Class 5 Accuracy for p=5: 80.83%
Class 6 Accuracy for p=5: 48.02%
Class 7 Accuracy for p=5: 49.03%
Class 8 Accuracy for p=5: 70.02%
Class 9 Accuracy for p=5: 74.63%
Overall Accuracy for p=10: 78.66%
Class 0 Accuracy for p=10: 95.10%
Class 1 Accuracy for p=10: 7.49%
Class 2 Accuracy for p=10: 95.54%
Class 3 Accuracy for p=10: 88.91%
Class 4 Accuracy for p=10: 82.59%
Class 5 Accuracy for p=10: 89.13%
Class 6 Accuracy for p=10: 89.14%
Class 7 Accuracy for p=10: 77.92%
Class 8 Accuracy for p=10: 85.93%
Class 9 Accuracy for p=10: 85.93%
Overall Accuracy for p=20: 82.03%
Class 0 Accuracy for p=20: 97.96%
Class 1 Accuracy for p=20: 0.00%
Class 2 Accuracy for p=20: 98.84%
Class 3 Accuracy for p=20: 94.85%
Class 4 Accuracy for p=20: 92.46%
Class 5 Accuracy for p=20: 95.40%
Class 6 Accuracy for p=20: 89.77%
Class 7 Accuracy for p=20: 80.16%
Class 8 Accuracy for p=20: 94.97%
Class 9 Accuracy for p=20: 88.90%
```