Assignment 2

CSE342 - Statistical Machine Learning

Ritika Thakur
2022408
CSAI

***Abstract*—Question 1: In question 1, we're tasked with implementing a Quadaratic Discriminant Analysis (QDA) classifier using the MNIST image dataset provided in the file mnist.npz. This dataset comprises 70,000 images, each with dimensions 28x28 and categorized into 10 classes. The dataset is split into a training set, containing 60,000 images, and a test set, containing 10,000 images. After loading the dataset using the NumPy library, we preprocess it and proceed to construct the QDA classifier. Subsequently, we evaluate the classifier's performance on the test set, calculating its accuracy, which we then print to the console.**

I. Methods

The code (SML_Ass2_Ques1) implements a Quadratic Discriminant Analysis (QDA) classifier for the MNIST dataset. Here's a summary of the methods used:

- **qda(train_images, train_labels, epsilon)**: This function trains the QDA model. It computes the mean and covariance matrices for each class in the training dataset. Regularization is applied to the covariance matrices to avoid singularity issues. The regularization parameter ϵ is passed as an argument.
- **categorise(test_images)**: This function categorizes test images using the trained QDA model. It calculates the probability of each class for a given test image and returns the class with the maximum probability.
- **accuracy(test_images, test_labels)**: This function calculates the accuracy of the QDA classifier on the test dataset. It compares the predicted labels with the true labels and computes the accuracy for each class as well as the overall accuracy.
- **Loading MNIST Dataset**: The MNIST dataset is loaded using TensorFlow's Keras API. It consists of training and test images along with their corresponding labels.
- **Plotting the Dataset**: The code includes a visualization of the MNIST dataset, showing five sample images for each class using Matplotlib.
- **Testing Different Regularization Parameters**: The code tests the QDA classifier with different values of the regularization parameter (ϵ). It evaluates the classifier's accuracy for each value of ϵ and plots the results.

II. QDA Expression

The QDA Expression used in the code is:

$$\begin{aligned}\text{prob} = \Bigg[& \log\left(\frac{\text{count}[i]}{\text{len(train\_images)}}\right) \\ & - 0.5 \times \text{log\_cov\_det}[i] \\ & - 0.5 \times (\text{test\_images} - \text{mean}[i])^T \, \text{cov\_inv}[i] \, (\text{test\_images} - \text{mean}[i]) \Bigg] \\ & \text{for} \quad i \in \{0, 1, 2, \ldots, 9\}\end{aligned} \tag{1}$$

III. Assumptions

- Due to the presence of singular matrix a small value of epsilon is added to the diagonal of the covariance matrix to make it non singular.
- The accuracy of the classifier varies for different values of epsilon.

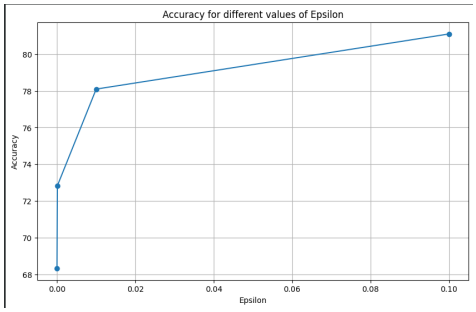

Fig. 1. Epsilon vs Accuracy

```
Accuracy of each class
0 --> 96.02 %
1 --> 93.744 %
2 --> 46.802 %
3 --> 51.188 %
4 --> 45.214 %
5 --> 29.596 %
6 --> 94.363 %
7 --> 43.288 %
8 --> 85.421 %
9 --> 93.162 %
Overall Accuracy:  68.34
Overall accuracy for epsilon=1e-06: 68.34
```

Fig. 2. Class-wise accuracy for different epsilons (1)

```
Accuracy of each class
0 --> 95.306 %
1 --> 93.656 %
2 --> 58.915 %
3 --> 61.485 %
4 --> 54.481 %
5 --> 38.229 %
6 --> 93.946 %
7 --> 48.93 %
8 --> 86.756 %
9 --> 92.468 %
Overall Accuracy:  72.83
Overall accuracy for epsilon=0.0001: 72.83
```

Fig. 3. Class-wise accuracy for different epsilons (2)

```
Accuracy of each class
0 --> 95.204 %
1 --> 93.656 %
2 --> 72.19 %
3 --> 71.683 %
4 --> 66.293 %
5 --> 47.87 %
6 --> 93.215 %
7 --> 57.879 %
8 --> 87.885 %
9 --> 91.378 %
Overall Accuracy:  78.09
Overall accuracy for epsilon=0.01: 78.09
```

Fig. 4. Class-wise accuracy for different epsilons (3)

IV. Plots

Images from each class:

***Abstract*—Question 2: Question 2 builds upon the concepts explored in Question 1, focusing on generating and visualizing reconstructed images from the MNIST dataset. This process involves utilizing a specific number of eigenvalues and employs Principal Component Analysis (PCA) along with Mean Squared Error (MSE) for the reconstruction task. Additionally, the accuracy of the reconstructed images compared to the test data is evaluated using the mnistClassifier from Question 1. Finally, the achieved accuracy is displayed in the console.**

```
Accuracy of each class
0 --> 95.0 %
1 --> 93.656 %
2 --> 77.713 %
3 --> 77.624 %
4 --> 72.403 %
5 --> 55.717 %
6 --> 93.111 %
7 --> 63.424 %
8 --> 88.398 %
9 --> 90.783 %
Overall Accuracy:  81.09
Overall accuracy for epsilon=0.1: 81.09
```

Fig. 5. Class-wise accuracy for different epsilons (4)

```
Epsilon: 1e-06, Accuracy: 68.34
Epsilon: 0.0001, Accuracy: 72.83
Epsilon: 0.01, Accuracy: 78.09
Epsilon: 0.1, Accuracy: 81.09
```

Fig. 6. Class-wise accuracy for different epsilons (5)

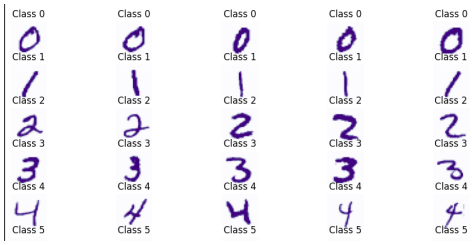

Fig. 7. Dataset (1)

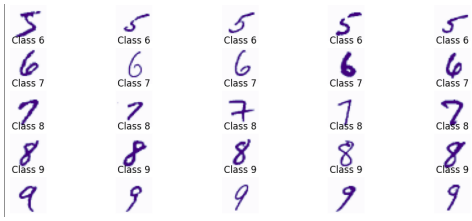

Fig. 8. Dataset (2)

V. Methods

The provided Python code preprocesses the MNIST dataset, applies Principal Component Analysis (PCA) for dimensionality reduction, reconstructs images, and evaluates the performance of the QDA classifier on the reduced dataset. Here's a summary of the code:

- **Loading the MNIST Dataset**: The MNIST dataset is loaded using TensorFlow's Keras API, and a visualization of the dataset is displayed.
- **Preprocessing**: The code preprocesses the training data by flattening the images into vectors, stacking them horizontally to form the data matrix, and creating a new label vector.
- **Applying PCA**: PCA is applied to reduce the dimensionality of the data. The code computes the covariance matrix, eigenvalues, and eigenvectors, sorts the eigenvectors, and selects principal components.
- **Performing Dimensionality Reduction and Reconstruction**: PCA is used to perform dimensionality reduction, and the original data is reconstructed from the reduced representation.
- **Plotting Reconstructed Images**: The code includes a function to plot reconstructed images using different numbers of principal components.
- **Calculating Accuracy**: Functions are provided to calculate the accuracy and class-wise accuracy of the QDA classifier for different numbers of principal components.
- **Applying QDA and Calculating Accuracy**: The code applies QDA to the reduced dataset and calculates accuracy for each number of principal components.
- **Plotting Accuracy**: The overall accuracy and accuracy for each class are plotted as a function of the number of principal components.

VI. Assumptions

- As the value of p increases the generated image becomes closer to the original image.
- Value of Mean Squared Error (MSE) is very close to 0. In our code it come out to be $6.58742705091612 \times 10^{-23}$

VII. Generated Images

5 generated images from each class for different 'p' are below:

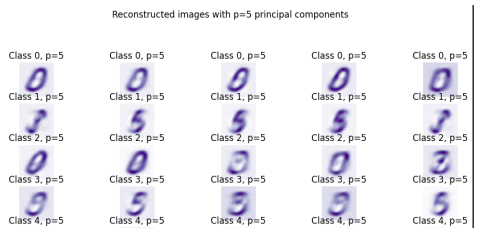

Fig. 9. p = 5 (1)

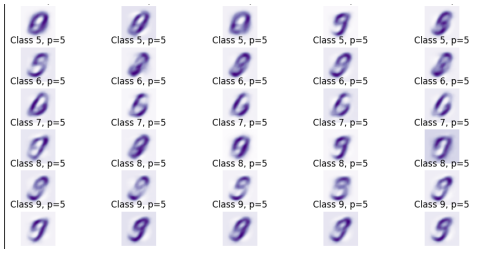

Fig. 10. p = 5 (2)

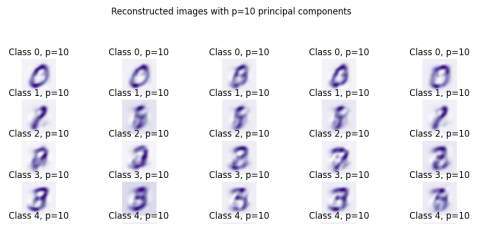

Fig. 11. p = 10 (1)

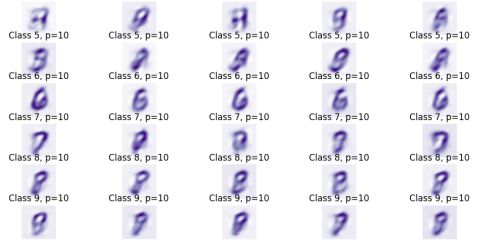

Fig. 12. p = 10 (2)

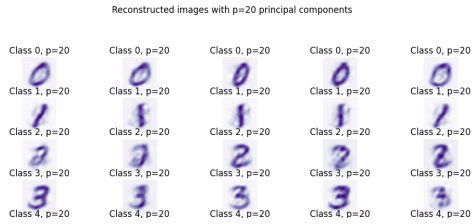

Fig. 13. p = 20 (1)

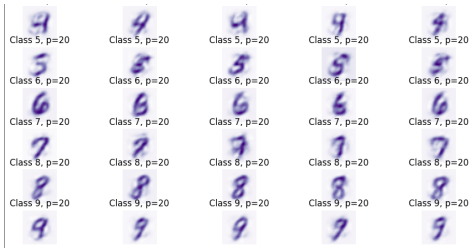

Fig. 14. p = 20 (2)

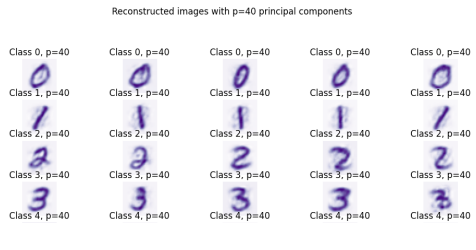

Fig. 15. p = 40 (1)

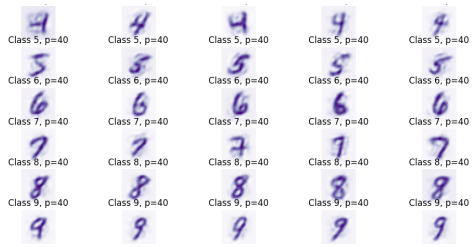

Fig. 16. p = 40 (2)

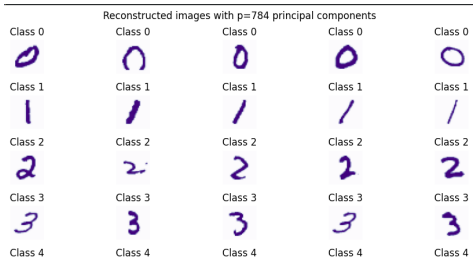

Fig. 17. p = 784 (1)

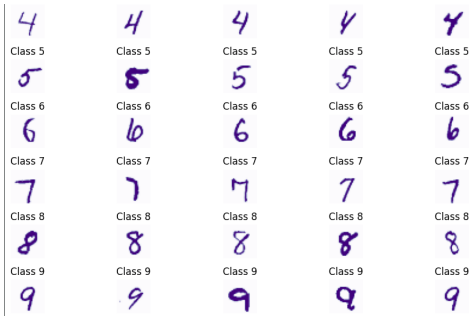

Fig. 18. p = 784 (2)

```
Accuracy of each class
0 --> 89.286 %
1 --> 94.273 %
2 --> 63.372 %
3 --> 71.485 %
4 --> 66.395 %
5 --> 50.785 %
6 --> 84.029 %
7 --> 76.751 %
8 --> 60.986 %
9 --> 55.5 %

Overall Accuracy
   71.74 %
```

Fig. 19. Accuracy for p = 5

```
Accuracy of each class
0 --> 95.102 %
1 --> 97.533 %
2 --> 90.891 %
3 --> 88.02 %
4 --> 84.623 %
5 --> 85.65 %
6 --> 94.572 %
7 --> 87.549 %
8 --> 82.341 %
9 --> 81.863 %

Overall Accuracy
   88.95 %
```

Fig. 20. Accuracy for p = 10

```
Accuracy of each class
0 --> 98.469 %
1 --> 97.093 %
2 --> 95.93 %
3 --> 94.653 %
4 --> 95.927 %
5 --> 94.395 %
6 --> 96.242 %
7 --> 91.926 %
8 --> 92.094 %
9 --> 92.765 %

Overall Accuracy
   94.97 %
```

Fig. 21. Accuracy for p = 20

```
Accuracy of each class
0 --> 98.98 %
1 --> 96.828 %
2 --> 96.899 %
3 --> 96.634 %
4 --> 98.371 %
5 --> 96.3 %
6 --> 96.764 %
7 --> 92.607 %
8 --> 94.969 %
9 --> 93.756 %

Overall Accuracy
   96.2 %
```

Fig. 22. Accuracy for p = 40

VIII. Accuracy Analysis

The accuracy for each class and overall accuracy for different values of p is given below:

Accuracy vs 'p' analysis for all classes in the form of line plots:

IX. Conclusion

Question 1 and Question 2 produce the expected results. In question 1 we have successfully implemented the expression for qda to classify the given dataset with an accuracy of upto 80%.

In question 2 we have successfully applied PCA to generate

images of the dataset for different datasets and observe an increase in accuracy with increase in 'p'.

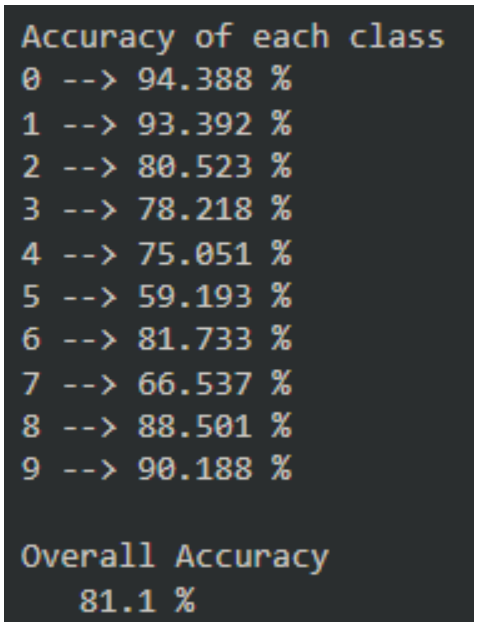

Fig. 23. 784

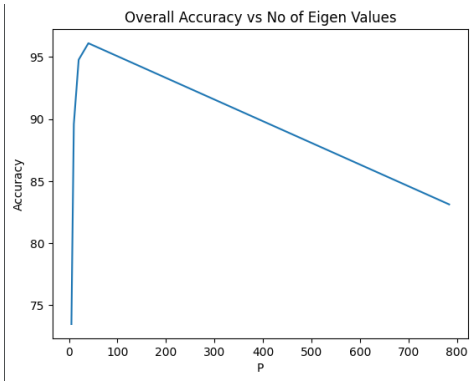

Fig. 24. Overall accuracy vs p

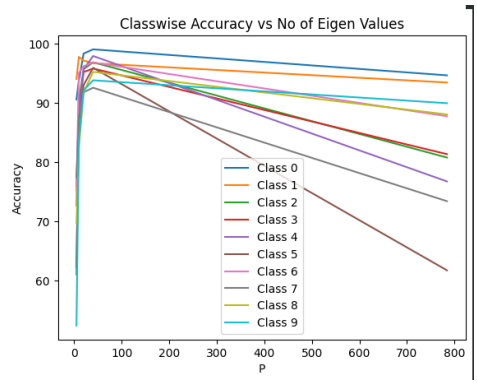

Fig. 25. Accuracy for each class vs p