CS464: Introduction to Machine Learning

Homework II

Deniz Aydemir
22001859
Section 1

December 13, 2023

Question 1

1.1

Here is the PVEs for the first 10 principal components I found:

PC 1: 0.097
PC 2: 0.071
PC 3: 0.062
PC 4: 0.054
PC 5: 0.049
PC 6: 0.043
PC 7: 0.033
PC 8: 0.029
PC 9: 0.028
PC 10: 0.024

and here is the histogram corresponding to:

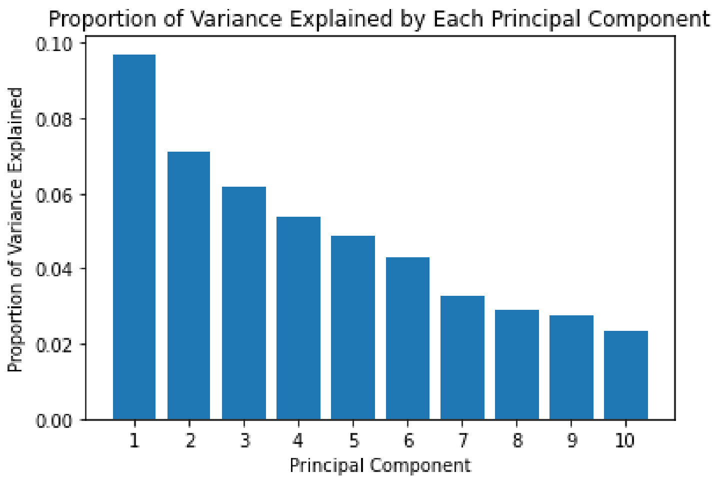


Figure 1: PVE Histogram for first 10 PCs

With summing the PVE values of first 10 PC's values up, we can say that they explain the 49 percent of the MNIST data.

1.2

In order to calculate the number of PCs required to explain the 70 percent of the data, I wrote (pca is a object of the PCA class I customly defined):

```
num_pcs = np.argmax(np.cumsum(pca.pve) >= 0.7) + 1
print(f"# of PCs required to explain the 70% of the data: {num_pcs}")
```

```
# of PCs required to explain the 70% of the data: 26
```

We need the first 26 PC's, and here is the corresponding graph:

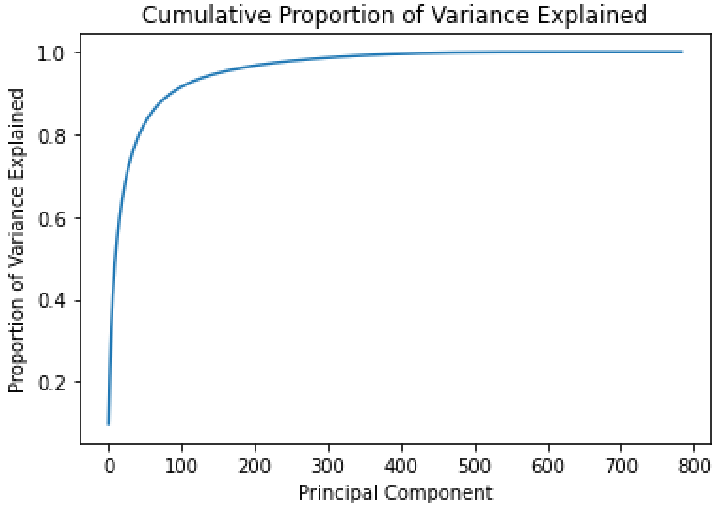


Figure 2: Cumulative summation of the PVE's

1.3

Here is the visualization of the first 10 PCs I found:

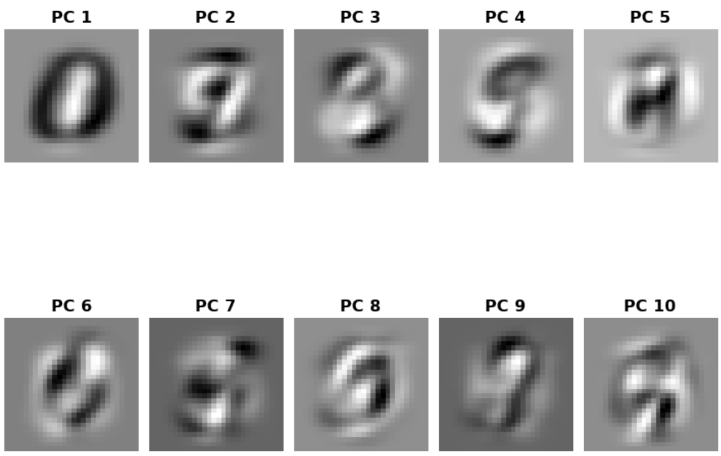


Figure 3: Visualization of the first 10 PCs

It seems the first principal component reflects to the shape of "0", biggest variation is captured in this shape. Other than that, I can see the shapes of "3" and "9" in the third and fourth principal components respectively. I think second principal component captured the horizontal upper line and inclined middle line of the numbers like "2", "4", "7", and "9" whereas principal component 5 kinda resembles the numbers "4" and "5". Moreover, principal component 6 kinda reflects the shape of "6" and principal component 9 kinda reflects the shape of "3". Lastly, in my opinion, principal component 10 reflects the shape of "5".

1.4

Here are the projection of the first 100 images of the dataset onto the first 2 principal components:

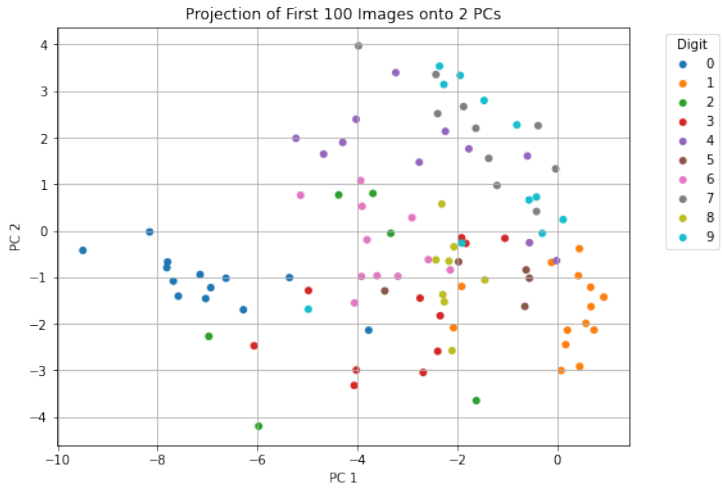


Figure 4: projection of the first 100 images onto the first 2 PCs

Biggest variation of principal component 1 is the "0" as expected since its visualization resembles the shape "0". Moreover, biggest variations of the principal component 2 seems like "2" and "7", but I'm not hundred percent sure. Since principal component 2 resembles to upper part of these numbers, as we saw in the Q 1.3, it can be expected. "1" has the least variance in the principal component 1, which resembles the shape "0", as its shape doesn't have any curves, and its a straight line in most cases. Moreover, clusters of "0" and "1" are distinguish-ably far away in the PC 1 axis. This means that our PC 1 can separate these classes well. For PC 2, it seems like "0" has the least variance, but it is normal since corresponding variance of PC2 is perpendicular to PC 1's. Moreover, it seems like PC 2, which captures a horizontal upper line and an inclined middle line, can distinguish "1", whose shape does not have any horizontal upper line and does not have any inclined middle line most of the cases, from the numbers like "2" and "7" whose shape has these features. In conclusion, we can expect that our multinomial logistic regression classifier will perform well, maybe the best, on the number "1" since the first two principal components can distinguish it well.

1.5

Note: To solve the question 1, I defined a class named PCA, and I used the MNIST data as follows:

```
class PCA:
    # some other codes
    def fit(self, images):
        X = images.copy()
        self.mean = X.mean(axis=0)
        X = X - self.mean
        # some other codes
```

Since I copied the data and subtracted the mean from the copied data, I didn't added the mean back to the original data in this question as stated in the hint.

We can sort the eigenvectors obtained from PCA according the their PVEs in a descending order. After we sort them, we can choose first certain "k" components among them. If we multiply the image with the first "k" eigenvectors, we would get a projection of the image on first "k" PCAs. After that, if we multiply the projection of the image with the transpose of the first "k" eigenvectors, we would get the reconstructed image. Below is an example:

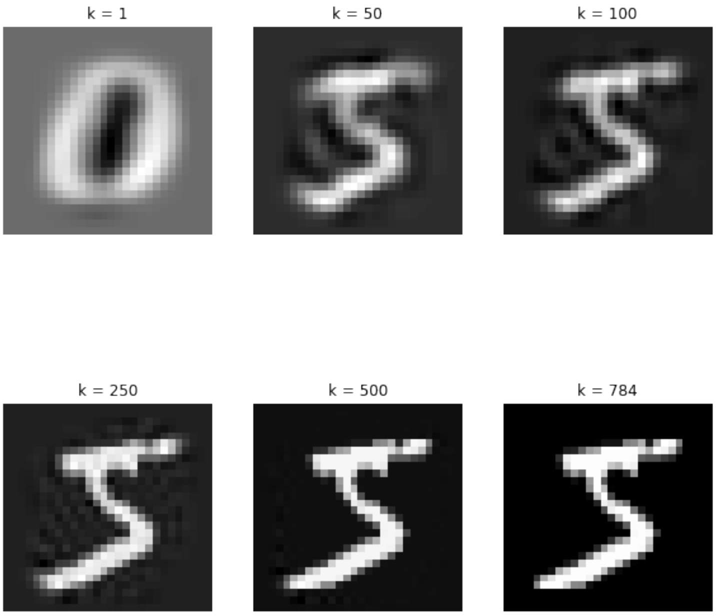


Figure 5: Reconstructing a MNIST image using first "k" eigenvectors

As we increase the parameter "k", the quality of the image increases. When we use only the first PC,reconstructed image is so similar to visualization of the PC 1 itself; and when we use all of the PCs , reconstructed image is same with the original image. However, in my opinion, one can transmit the information stored in the original image (a number in our case) by reconstructing it on the first "k" eigenvectors. This way, transmitted data can be shrinked to nearly 1/7 of its original size.

Question 2

2.1

Text accuracy for the base model: 0.9062

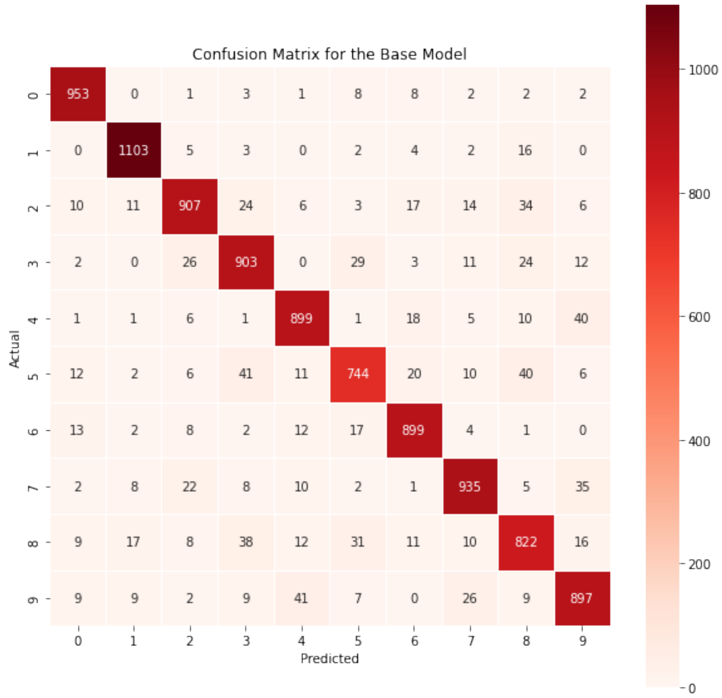


Figure 6: Confusion Matrix for the Base Model

2.2

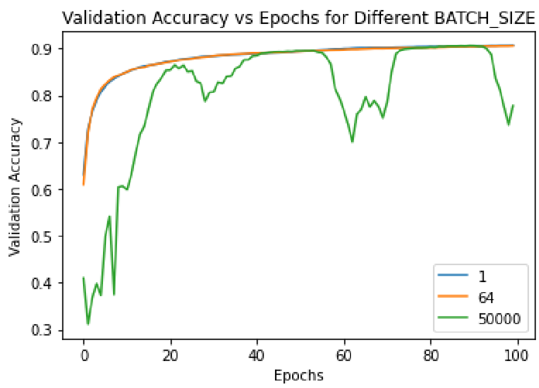


Figure 7: Validation Accuracy vs Epochs for Different Batch Sizes

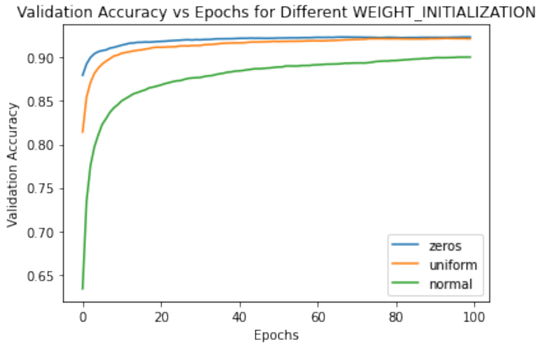


Figure 8: Validation Accuracy vs Epochs for Different Weight Initializations

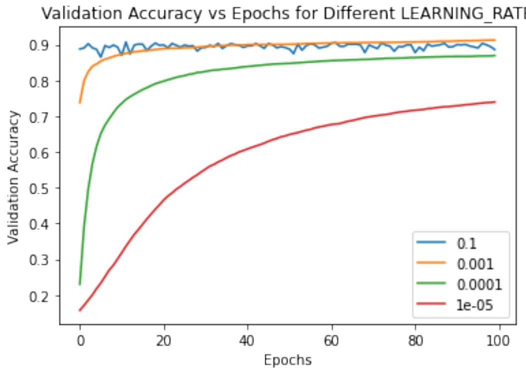


Figure 9: Validation Accuracy vs Epochs for Different Learning Rates

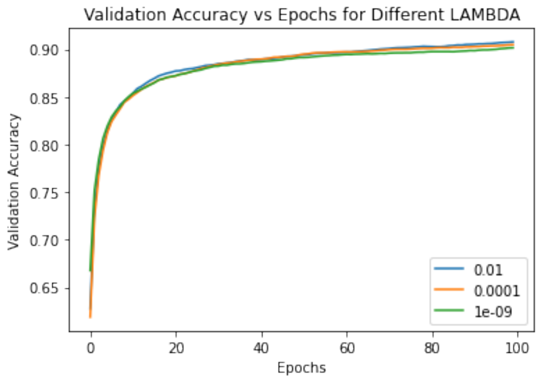


Figure 10: Validation Accuracy vs Epochs for Different Lambdas

2.3

Even though there is 0.09 difference among the accuracies of the batch size 1 and 64, I chose batch size as 64 to decrease the execution time. It decreased to 1 minute from 10 minutes. Here is the best parameters:

- Batch Size: 64
- Weight Initialization: zeros
- Learning Rate: 0.001
- Lambda: 0.01

Test accuracy for the best model: 0.9259

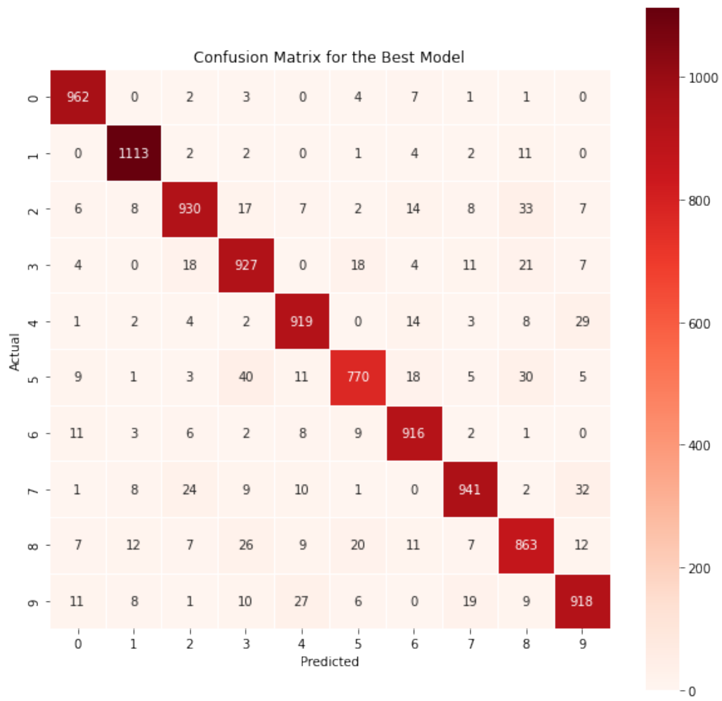
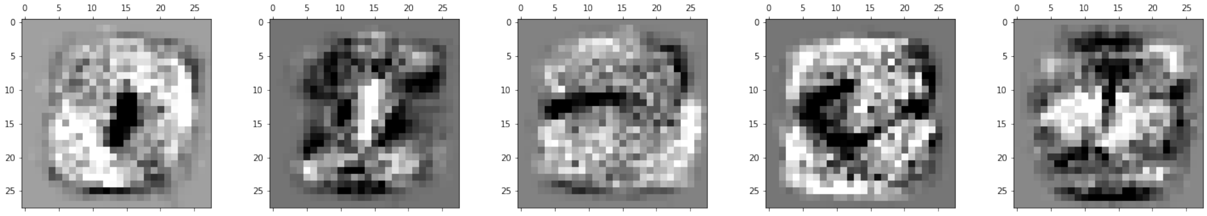
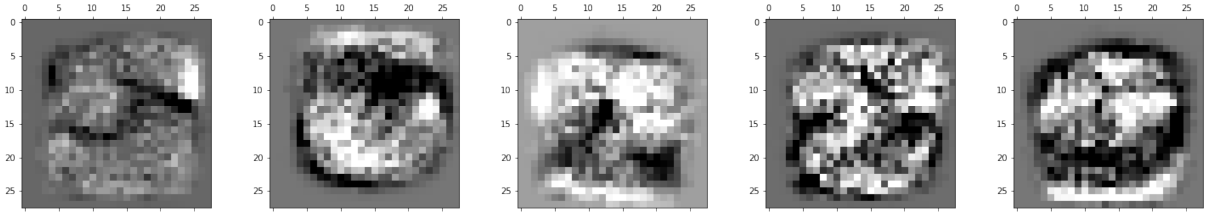


Figure 11: Confusion Matrix for the Best Model

2.4



(a) First Weight (b) Second W. (c) Third Weight (d) Fourth W. (e) Fifth Weight



(f) Sixth Weight (g) Seventh W. (h) Eighth Weight (i) Ninth Weight (j) Tenth Weight

Figure 12: Images of Weights

First weight resembles the circle-like shape of a "0". Second weight kind of captures the straight line in the middle of a "1", whereas third weight reflects the shape of a "2" even though lower part of it is highly blurry. Fourth weight captures the dent in the right part of a "3". I know that fifth weight is corresponding to number "4", however, I couldn't infer anything from it. Sixth weight resembles a "5", and seventh weight captures the bottom and upper left part of a "6". Even tough its bottom part is blurry, but eight weight captures the upper - upper middle part of a "7". Ninth weight resembles the shape of an "8" and tenth weight captures the upper part of a "9".

2.5

```
--------------------------------------------------
Precision for number 0: 0.950592885375494
Recall for number 0: 0.9816326530612245
F1 Score for number 0: 0.965863453815261
F2 Score for number 0: 0.9752635847526357
--------------------------------------------------
Precision for number 1: 0.9636363636363636
Recall for number 1: 0.9806167400881057
F1 Score for number 1: 0.9720524017467249
F2 Score for number 1: 0.9771729587357332
--------------------------------------------------
Precision for number 2: 0.9327983951855566
Recall for number 2: 0.9011627906976745
F1 Score for number 2: 0.916707737801873
F2 Score for number 2: 0.9073170731707318
--------------------------------------------------
Precision for number 3: 0.8930635838150289
Recall for number 3: 0.9178217821782179
F1 Score for number 3: 0.9052734375000001
```

```
F2 Score for number 3: 0.9127609294998031
--------------------------------------------------
Precision for number 4: 0.9273461150353178
Recall for number 4: 0.9358452138492872
F1 Score for number 4: 0.9315762797769894
F2 Score for number 4: 0.9341329538524091
--------------------------------------------------
Precision for number 5: 0.9265944645006017
Recall for number 5: 0.8632286995515696
F1 Score for number 5: 0.893789901334881
F2 Score for number 5: 0.8751989088429187
--------------------------------------------------
Precision for number 6: 0.9271255060728745
Recall for number 6: 0.9561586638830898
F1 Score for number 6: 0.9414182939362796
F2 Score for number 6: 0.950207468879668
--------------------------------------------------
Precision for number 7: 0.9419419419419419
Recall for number 7: 0.9153696498054474
F1 Score for number 7: 0.9284657128761716
F2 Score for number 7: 0.9205634905106632
--------------------------------------------------
Precision for number 8: 0.881511746680286
Recall for number 8: 0.8860369609856262
F1 Score for number 8: 0.883768561187916
F2 Score for number 8: 0.8851282051282051
--------------------------------------------------
Precision for number 9: 0.9089108910891089
Recall for number 9: 0.9098116947472745
F1 Score for number 9: 0.9093610698365527
F2 Score for number 9: 0.9096313912009513
--------------------------------------------------
```

We know that from Q 2.3, best model's confusion matrix's best result is the result of the number ”1”. Moreover, from Q 1.4, we also know that both first and second principal component distinguishes ”1”s well. Therefore, it is not a surprise (actually confusion matrix directly indicates that) best F scores belongs to ”1”. Moreover, second best F scores are belongs to ”0”. It is not a surprise since we know that ”0” has the highest variance on the first principal component from Q 1.4 and it clearly resembles the shape of a ”0” as we know from Q 1.3. Lastly, worst F scores, 0.87 and 0.89, belongs to ”5”.