1) Coding practice [10 pts]

There are two problems in this coding practice. In the first problem, you will study how different numbers of principal components represent the images visually. For the second problem, you will use logistic regression to predict the class label of images using the principal components representation of the images and examine how the classification error changes with the number of principal components used. We will explore the MNIST handwriting digits loaded from tensorflow. More details and hints are available in 'HW4.ipynb.'

1. a) PCA for dimension reduction (3pt)

(a) For k = 0, 10, 20, 30, 40, 49, use k-th principal components ONLY, namely,

$$\hat{x}_i = \overline{X} + (\varphi_k^\top x_i)\varphi_k$$

, where x_i is the reconstructed image, $\overline{X}$ is the averaged image, and φ_k is the k-th principle axis for **MNIST 0**'s to approximately reconstruct the image selected above. Note that we index from 0, namely 0-th principal component is the first one. Display the reconstruction for each value of k. To display the set of images compactly, you may want to use the 'plot images' function defined in 'HW3.ipynb.'

```
from sklearn.decomposition import PCA
height = 28
width = 28
num_components = 50
k = [0, 10, 20, 30, 40, 49]
pca = PCA(num_components).fit(x)
pcs = pca.fit_transform(x)
principal_vectors = pca.components_
principal_vectors = principal_vectors.reshape((num_components, height, width))
principal_components = pca.components_
recon_img = []
k_values = [0, 10, 20, 30, 40, 49]
average_image = pca.mean_
recon_img = []
for k in [0, 10, 20, 30, 40, 49]:
    image_projection = np.dot(x[my_image], principal_components[k]) *
principal_components[k]
    recon_img_k = average_image + image_projection
    recon_img.append(recon_img_k.reshape(height, width))
labels = ['principal vector %d' % (i+1) for i in np.arange(num_components)]

plot_images(recon_img, labels, height, width, 2, 3)
```

From the according code above, we get the Variance recorded as 88.04%.

[Images of reconstruction based on each k-th component]

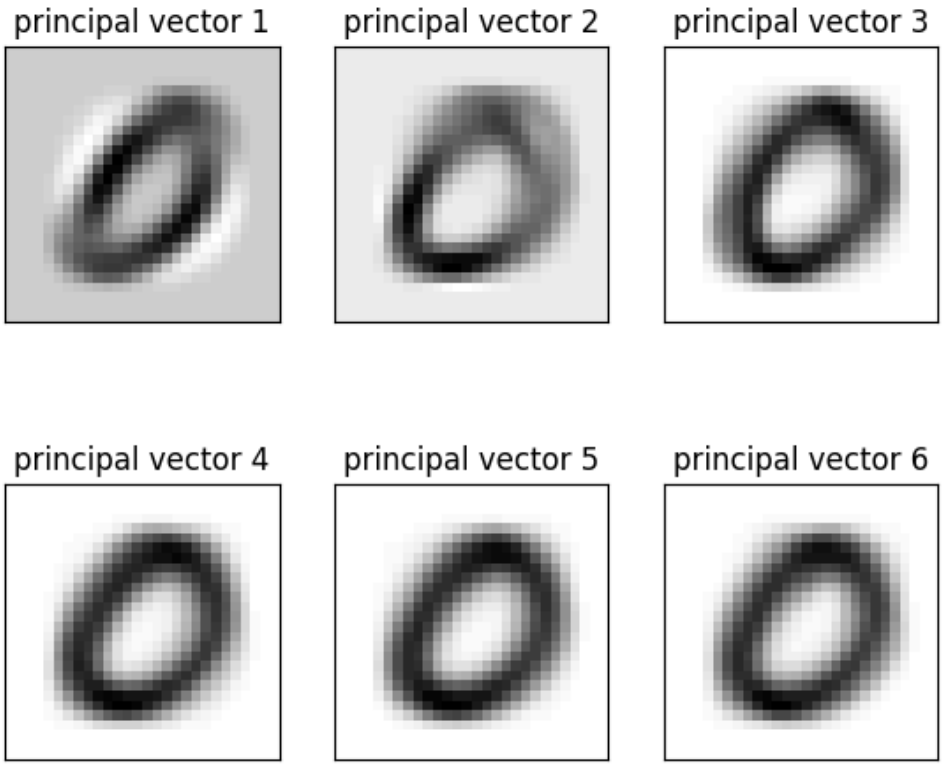

Figure 1: plot for tables

PCA components decomposes images into components based on the variance they capture from the original dataset. The first few principal components seize the most significant, general features. These components are broad, capturing features common to all images (namely the image '0' in this case). As we progress to higher-order components (the k value increasing) they focus on more unique features that define individual differences. The sequence of images shows the effect of using individual principal components. Early components are less defined because they're broad and general, in contrast, later components are more detailed. And this is due to the decrease of variance as the k increases. In other words, while they capture less variance as k increases, the features captured seem to be more specific and provide a clearer image.

1. b) PCA for classification (7 Pts)

(a) Load in the MNIST data with the labels as y and the images as x by running the next cell in the notebook. Create a subset of the data by keeping only the images that have the label of either 4 or 9. Use Principal Compo- nents Analysis (PCA) to project the data onto the first two principal components, and create a plot of the projected data color-coded by the label. Does the plot make sense? Explain in a couple of sentences. You don't need to do image reconstruction for this question.

```
indices = np.where((y == 4) | (y == 9))

x_sub = x[indices]
y_sub = y[indices]

pca = PCA(n_components=2).fit_transform(x_sub)

plt.figure(figsize=(10, 7))
for label, color in zip([4, 9], ['blue', 'red']):
    plt.scatter(pca[y_sub == label, 0], pca[y_sub == label, 1], c=color, alpha=0.5,
label=f"Digit {label}")
```

```
plt.title('PCA projection of digits 4 and 9')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.show()
```

[Plot]

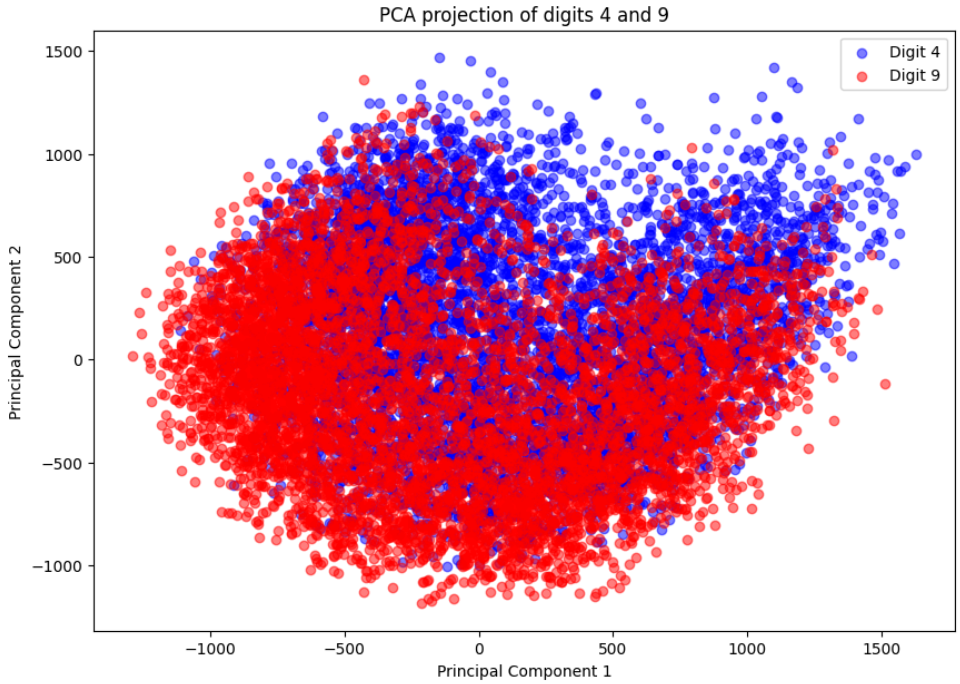

Figure 2: plot for tables

This revised explanation builds on the observation that there is significant overlap between digits 4 and 9 when plotted against the first and second principal components derived from PCA. This indicates that within the two-dimensional space formed by these principal components, there isn't a stark contrast that separates the two digits. The similarity in their structure, particularly the loops found in both digits and the extending lines that might be captured as primary variances by PCA, would account for this overlap.

Considering the shape of 4 and 9, which can be geometrically and stylistically similar, we would indeed anticipate that the most significant sources of variance could include features common to both numbers. If the looping structures or the line extensions are dominant in the first two principal components, this would naturally lead to the observed overlap. The second principal component seems to offer a slightly clearer separation, suggesting that it captures an aspect of the data that better differentiates between the digits, although not perfectly. This nuance indicates that while PCA provides some insights into the underlying structure of the data, additional or alternative methods might be necessary to achieve a clearer distinction between such closely related classes.

(b) Why not use more principal components? For k = 2, 3, 4, …, 15, use PCA to project the data onto k principal components. For each k, you will end up with k dimensional representation for each data point. Then use the k dimensional representations and logistic regression to build a model to classify images as 4 or 9, and calculate the accuracy of the model. Create a plot of accuracy as a function of k, the number of principal components used. Does the plot make sense? Explain in a few sentences. You don't need to do reconstruction for this question. Note: you need to report the accuracy on the test set, not the training set.

```
# Your Code Here
from sklearn.model_selection import train_test_split

def filtering(x, y) :
  idx = np.where((y == 4) | (y == 9))
  return x[idx], y[idx]

x_sub, y_sub = filtering(x,y)
x_test_sub, y_test_sub = filtering(x_test, y_test)

k_values = [k for k in range(2, 16)]

accuracies = []

for k in k_values :
  pca = PCA(n_components=k)
  x_train_pca = pca.fit_transform(x_sub)
  x_test_pca = pca.transform(x_test_sub)

  l_model = LogisticRegression(max_iter = 1000, solver='lbfgs')
  l_model.fit(x_train_pca, y_sub)

  accuracy = l_model.score(x_test_pca, y_test_sub)
  accuracies.append(accuracy)


plt.figure(figsize=(10, 6))
plt.plot(k_values, accuracies, marker='o')
plt.title('Model Accuracy vs Number of Principal Components')
plt.xlabel('Number of Principal Components')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()
```

[Plot]

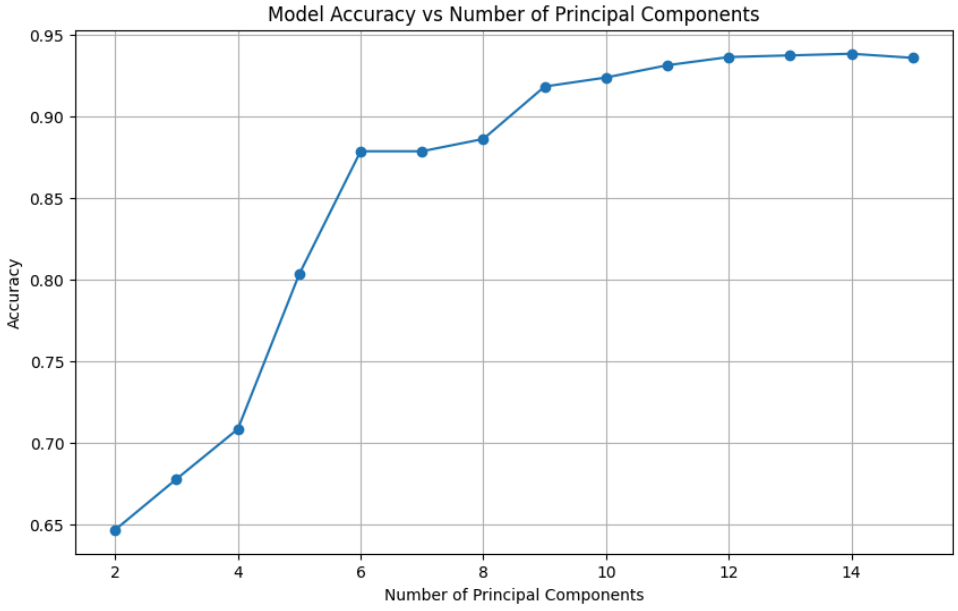

Figure 3: plot for tables

The plot makes sense as it shows the diminishing nature of adding more principal components in a logistic regression model for MNIST digit classification. Initially, as we increase the amount of componenets, the model's accuracy improves significantly because we capture more of the data's variance that distinguishes the two digits. However, if we keep on later components, the accuracy no longer improves. This suggests that the most critical information resides in the earlier components, and the later components do not contain more discriminative information of the two digits.

2) Multiple Choice Questions [2 pts]

(a) Given a set of seven one-dimensional data points {1, 2, 3, 4, 5, 6, 7}, suppose we run the K-Means algorithm to cluster them into two groups. The initial cluster centers are set at 1.8 and 2.8. Determine the final cluster centers after the K-Means algorithm converges. Please include your intermediate derivations leading to the final answer.

(A) 1.5 and 5.0

(B) 2.0 and 5.5

(C) 2.5 and 6.0

(D) 3.0 and 5.0

[explanation]

Step 0 : 1.8 - (1,2) , 2.8 - (3,4,5,6,7)

The mean of the first cluster is 1.5 and the mean of the second cluster is 5

Step 1 : 1.5 - (1, 2, 3), 5 - (4,5,6,7)

The new mean of the first cluster becomes 2 and the second is 5.5

Step 2 : 2 - (1, 2, 3), 5.5 - (4, 5, 6, 7)

The means for each cluster is maintained the same to 2 and 5.5 meaning that the K-Means algorithm now converges. Hence, the final result is 2.0 and 5.5

(b) Determine which of the following statements about PCA are true.

(A) PCA is a deterministic algorithm, and it will always produce the same results for the same input data.

(B) The first principal component captures the least variance in the data.

(C) PCA is a linear technique for dimensionality reduction.

(D) PCA is only applicable to categorical data.