BLG 454E Learning From Data (Spring 2018)

Homework 3

Onat Sahin - 150150129

1 Question 1

The code for this question can be found in pca.py .

a) Main motivations for reducing a dataset's dimensionality are eliminating erroneus features and visualizing the data.

b)We can train two seperate classifiers with the original data and the reduced data. Then we can compare the test results of both classifiers to see if the reduced data retained the important features.

c) It seems like the 2d data is projected on to the y-axis. However, it is mainly the x coordinate that determines the class. Therefore, performance of the given PCA is low. The data should be projected on to the x-axis.

d) There are several drawbacks of PCA. First of all, PCA assumes that the principle components of the given data are linear combinations of the original features and these principle components are orthagonal. These assumptions are not necessarily true all the time. If they are false, PCA fails to give an accurate representation of the original data. Also, PCA uses variance as a measurement of the importance of a feature, which may not be the case. Finally, if the number of dimensions the data is going to be reduced to is not carefully selected, some important features may be lost and PCA may not give an accurate representation of the original data.

e) Figure 1 shows my implementation of pca, which is included in "pca.py". Figure 2 shows the output of "pca.py". For some reason I could not understand, my output is the flipped version of the example output given in the question.

```python
#Onat Şahin 150150129
import numpy as np
import matplotlib.pyplot as plt
import random

def readData(dataFile): #Funtion to read the data to a list
    with open(dataFile) as file:
        list = [line.strip('\n') for line in file.readlines()]
    list = [line.split(',') for line in list]
    for instance in list:
        for i in range(len(instance)):
            instance[i] = int(instance[i])

    features = []
    labels = []

    for instance in list: #Remove the labels from features to a seperate list
        features.append(instance[0:len(instance)-1])
        labels.append(instance[-1])

    return features, labels

features, labels = readData('data.txt')
covariance = np.cov(features, rowvar=False) #Finding covariance matrix of features
eigenvals, eigenvecs = np.linalg.eig(covariance) #Finding eigenvalues and eigenvectors of the covariance matrix
index = eigenvals.argsort()[::-1] #Sorting eigenvalues in descending order
eigenvals = eigenvals[index]
eigenvecs = eigenvecs[:,index] #Sorting eigenvectors according to corresponding eigenvalues

reduced = []
for vec in eigenvecs:
    reduced.append(vec[0:2]) #Take the first two eigenvectors to reduce the data to two dimensions.

reduced = np.array(reduced)
reduced = reduced.transpose()
npfeatures = np.array(features)

x = []
y = []

for i in range(len(features)):
    temp = np.dot(reduced, npfeatures[i].transpose()) #creating the new features
    x.append(temp[0])
    y.append(temp[1])

plt.scatter(x,y,s=1)

samples = random.sample(range(0,len(x)), 200)

for i in samples:
    plt.annotate(labels[i], (x[i],y[i]))

plt.show()
```

Figure 1: My PCA implementation

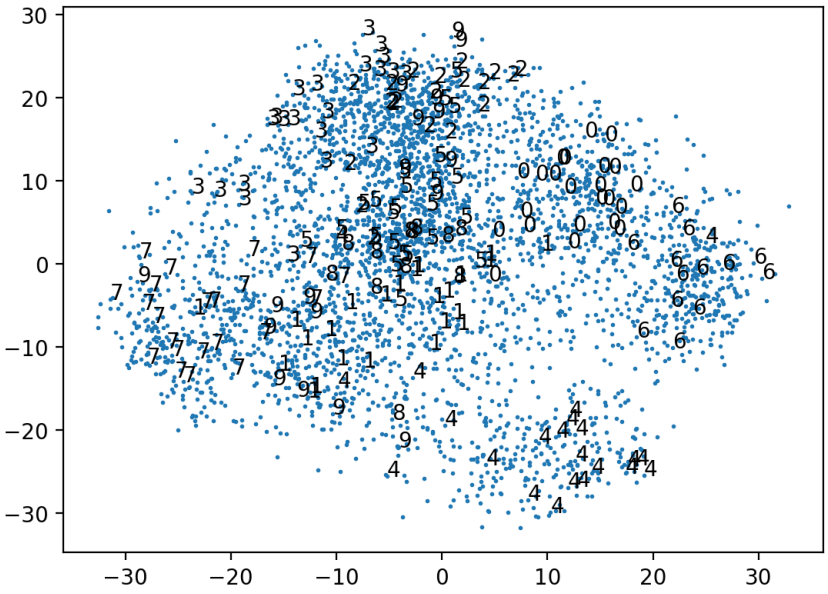

Figure 2: Output of my PCA implementation

2 Question 2

For this implementation, I assumed the term(rank) to be the percentage of the singular values used for the compressed image. For example, image obtained using 20 terms means the largest %20 of the singular values of the original image are used to obtain the output image. The outputs I got are not as clear as the example outputs given in the question because of the difference between the eigenvalue calculations of numpy "eig" function and the built-in svd function. Figure 3 shows the original while Figure 4, Figure 5, Figure 6 and Figure 7 shows the images obtained using 1, 5, 20 and 50 terms respectively.

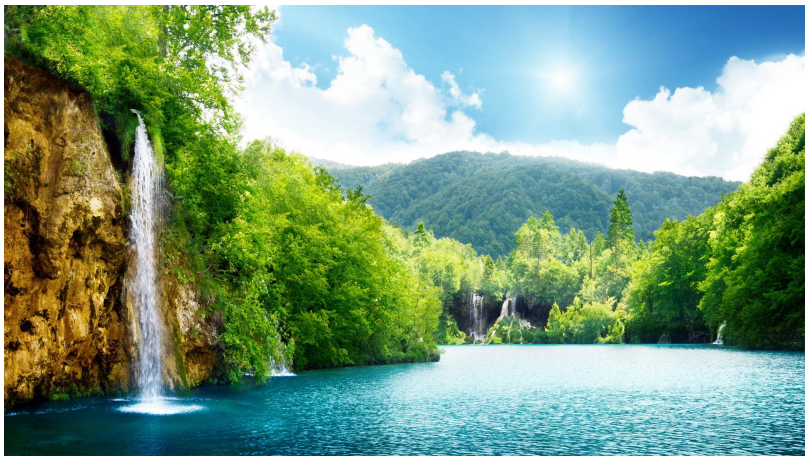

Figure 3: Original image

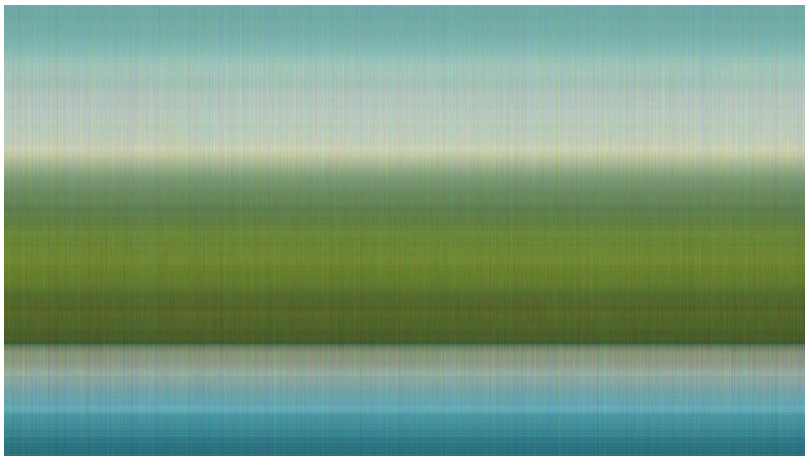

Figure 4: Image obtained using 1 term

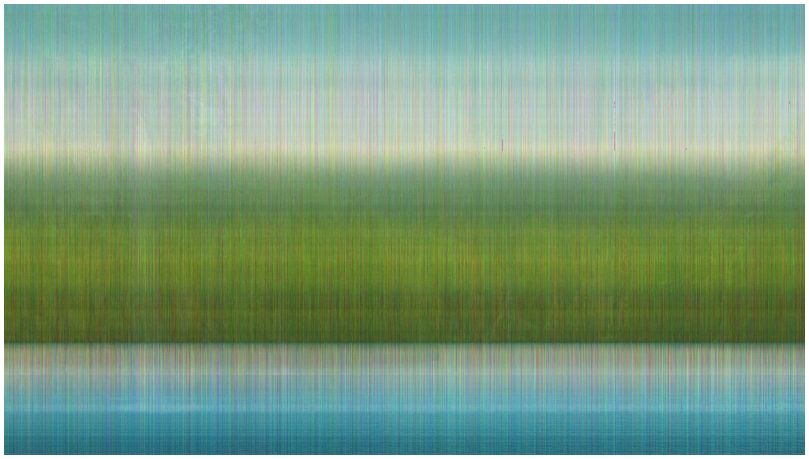

Figure 5: Image obtained using 5 terms

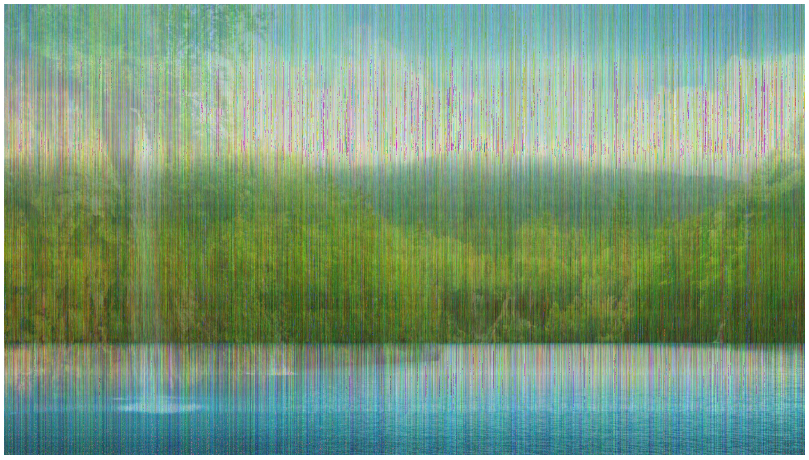

Figure 6: Image obtained using 20 terms

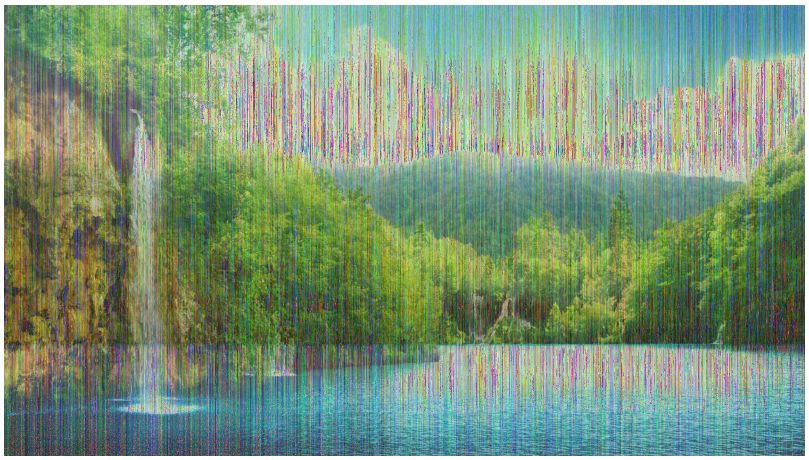

Figure 7: Image obtained using 50 terms