Technique Assignment 4: Principal Component Analysis

Cogs 109 Spring 2020
Due: May 22 11:59pm
100 points

Submit your completed assignment on Gradescope as a pdf report containing all code, graphs, and answers to questions. Include comments for clarity.

Section 1: Iris data (75 points)

Load Fisher's iris data set in Python from sklearn datasets using the command `iris = datasets.load_iris()`, then taking the transpose of the `iris.data` array. We will refer to the 4 x 150 dataset as `irisInputs`.

1. **(10 points)** Using a 2 x 2 subplot, **plot** a histogram for each row of `irisInputs`. Which dimension (row) has the greatest variance?

2. **(5 points)** Find and **report** (print) the mean of the dataset. This should be a 1x4 row with a mean value for each dimension in the dataset.

3. **(10 points)** Create a zero-meaned data matrix Z. You can do this by subtracting the mean column from every column of `irisInputs`. Hint: you can create a matrix that replicates the mean n times using `np.tile()`.
   Report your **code only** (do not output Z or the mean matrix).

4. **(5 points)** Calculate the covariance matrix of Z (You can do this by multiplying every element of Z by every other element of Z, then dividing by the number of samples – 1).

   ```
   C = np.matmul(Z,Z.T)/(n-1)
   ```

   Report the size of your covariance matrix C. Is the size correct?

5. **(10 points)** Using the covariance matrix C, answer these questions:
   a. Which dimension has the greatest variance?
   b. Which 2 dimensions are the most positively correlated?
   c. Which 2 dimensions are the most negatively correlated?
   d. Which 2 dimensions are the least correlated?

6. **(5 points)** Calculate the eigenvectors (V) and eigenvalues (D) of C using the Numpy linalg function `eig()`.

   ```
   D,V = np.linalg.eig(C)
   ```

Each column of V is an eigenvector of C.

D is a diagonal matrix. Every element along the diagonal is an eigenvalue corresponding to an eigenvector in V.

Sort the eigenvectors based on the magnitude of their corresponding eigenvalues. You may write your own code to sort the eigenvectors or use this provided code to create Vs, the matrix of sorted eigenvectors:

```
idx = D.argsort()[::-1]
Vs = V[:,idx]
```

Your eigenvectors may already be in the correct order before sorting, but they are not guaranteed to be ordered, so this is an important step to remember.

The sorted eigenvectors of the covariance matrix, Vs, are the components along which you will project your data. If you didn't want to reduce the dimensionality of your data, you would use every eigenvector for the next step. If you do want to reduce the dimensionality of your data (that's the point of PCA!), you will keep only the principal components, or the first k eigenvectors.

7. **(5 points)** Project your data into the new component space using only the first 2 columns of the transpose of Vs (the transpose is equal to the inverse).
   ```
   Proj = np.matmul(Vs[:,0:2].T,Z)
   ```

8. **(10 points)** Plot the projected data in component space (plotting the first row of Proj along Component 1 and the second row of Proj along Component 2).

   Next, to reconstruct the data in terms of your original dimensions, you will multiply your projected data by your principal components. Finally, you will add in the mean that you subtracted earlier.

9. **(15 points)** Reconstruct your data back into the original 4-dimensional coordinate space.
   ```
   ReconstData = np.matmul(Vs[:,0:2],Proj) + dataMean;
   ```

   Does your reconstructed data look similar to your original data? Explain why or why not, using a plot or plots to support your answer.

Section 2: Face data (25 points)

First, download the file “faces_40x40_500.csv” from Canvas. We will refer to the data as `facemat`.

This file consists of a 1600x500 matrix, storing 500 faces of 100 celebrities, 5 faces from each. Each face is an image of size 40x40 and we have reshaped each image into a **column** vector of length 1600. Although each face image is viewed as a 2D matrix, we need to think of it as a vector since each face is technically a single data sample/instance. The function `reshape()` can be used to convert between column vector and 40x40 matrix.

In short, `facemat` is a 1600x500 matrix with each column storing one face. Show your code and images generated to answer each question.

10. **(5 points)** Show the first 25 faces by generating a figure of 5 x 5 subplots with each subplot displaying a single face. You may want to use a loop for this portion also. Use `reshape()` to convert the 1600 pixels into a 40x40 matrix. Use `imshow()` to plot using grayscale. You may want to put your code into a loop that runs 25 times.

11. Calculate the mean (average) face of all 500 faces and display the mean face (You can compute the mean of all vectors using `mean()` and then use reshape on your mean vector to transform it into a 40x40 image).

    Now we will perform PCA on the face data step by step.

12. Create a zero-mean matrix Z from `facemat`. You can do this by subtracting the mean column from every column of `facemat`.

13. Calculate the covariance matrix of Z (You can do this by multiplying every element of Z by every other element of Z, then dividing by the number of samples – 1).
    Report the size of your covariance matrix. Is the size correct?

14. Calculate the eigenvectors (V) and eigenvalues (D) of C using the Numpy function `eigh()`. **Note: `eigh()` will work similarly to `eig()` but will avoid complex eigenvectors produced by rounding errors.**
    This may take a little while. You may want to include `print("Done")` after this line of your code so you know when it is completed.

    Each column of V is an eigenvector of C.
    D is a diagonal matrix. Every element along the diagonal is an eigenvalue corresponding to an eigenvector in V.

15. **(5 points)** Sort the eigenvectors based on the magnitude of their corresponding eigenvalues.

16. **(5 points)** Display the top 25 eigen-faces, which should be the first 25 columns of Vs, the matrix of eigenvectors. Show each face in a subplot in a 5x5 grid. Hint: Use `reshape()` and `imshow()` as before.

17. **(10 points)** Reconstruct the faces using varying numbers of principal components. Please create a figure with 1 by 4 subplots, as follows:

    a. Use the top 20 principal components to reconstruct the first face in facemat. Display the reconstructed face in subplot 1.
    b. Repeat (a) using the top 40 components. Display the reconstructed face in subplot 2.
    c. Repeat (a) using the top 80 components. Display the reconstructed face in subplot 3.
    d. Repeat (a) using the top 120 components. Display the reconstructed face in subplot 4.

    Report your observation and provide a brief explanation of your data.

18. Extra credit (up to 10 points): Download puffin.csv from Canvas. This is a 1600x1 image. You can reshape the vector into a 40x40 image and plot in grayscale. Using the components you generated above (top 120 components), transform this image by subtracting out the mean face, projecting the image along the first 120 components, then reconstructing the image in pixel space. Plot the original and the reconstructed image. Explain your result.