**Assignment 1**

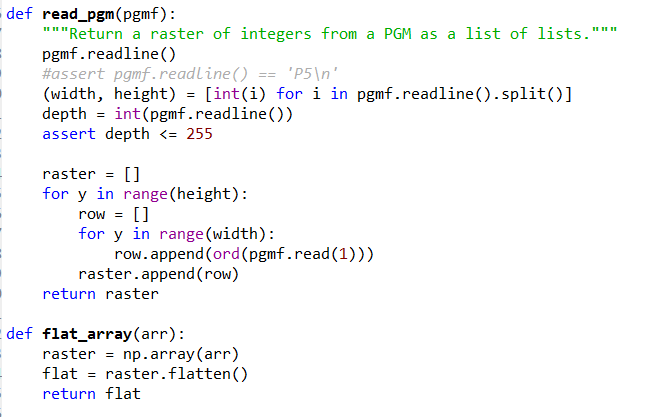
This is a report that goes through the steps of completing Assignment 1

Given a data set of pictures for people from

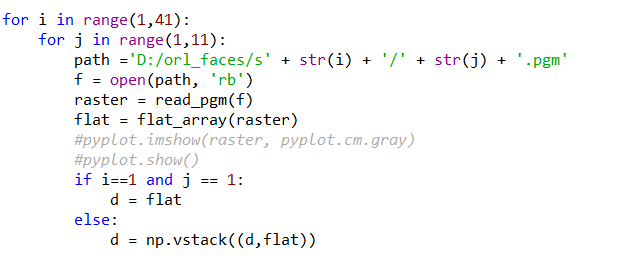
https://github.com/marinkaz/nimfa/tree/master/nimfa/datasets/ORL\_faces

**Step 1: Understanding and performing the preprocessing step on the Data (First 20 points)**

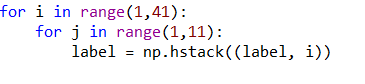
This involved knowing more about the data type PGM, and how to read it via python. We used a function that reads the image into a 2D 112x92 array, then wrote a [flat\_array(raster)] function that takes in the 2D array and returns a 10304x1 vector.



Then we appended all the vectors from the dataset, using variables to go through different path names, into a 400x10304 set using vstack.



At the same time, we initiated a 1x40 vector of 10 1’s, 10 2’s… etc.



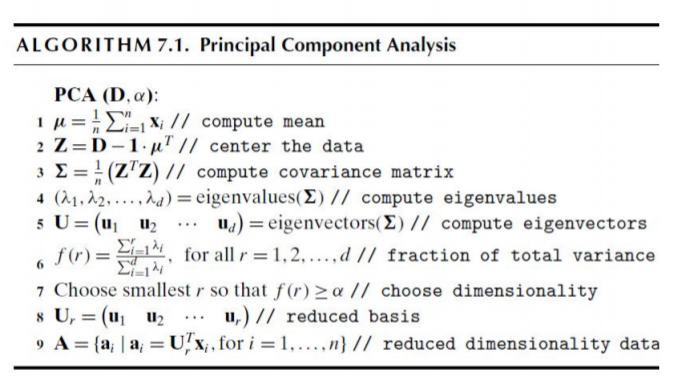
**Step 2: Splitting the data**

Simply enough, we perform the following index operation d[0::2] & d[1::2] on both the Data and the Label arrays, resulting in 4 arrays: Training data, Training label, Testing Data, Testing Label

**Step 3: PCA**

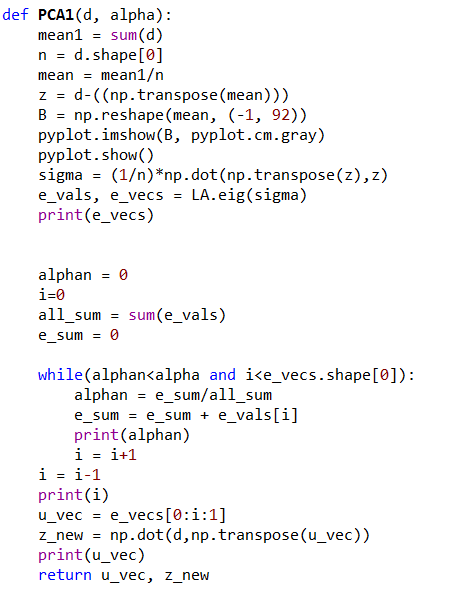
The challenging part of PCA was the running time, it takes around 20 minutes so coding can often get frustrating.

**a. We performed the PCA algorithm according to the given pseudo code**



As this is a very time-consuming step, a part of our code involved using ready-made PCA to make sure the other functions work correctly a bit faster.

**PCA1 (PCA by the given Pseudo code)**



(the other PCA2 function is the one we used for faster debugging)

**b. Function call**

The input is the training data and the desired alpha

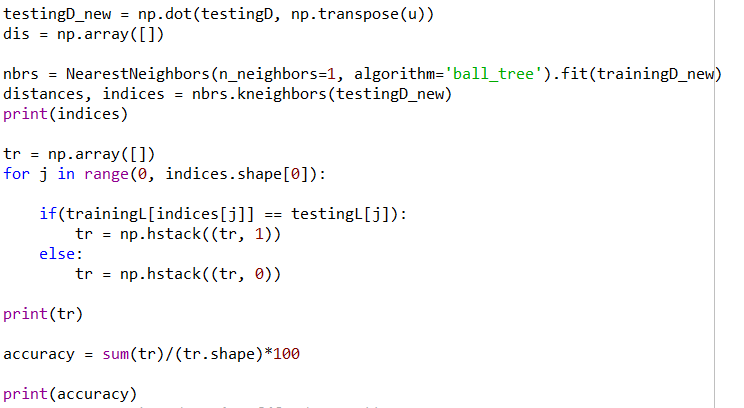


**c. Testing and computing accuracy**

Here, we take the output Training data and U vectors, we perform the same operation on the testing data and then apply the K-Nearest Neighbors from SKLearn.

If the result label from KNN of each sample is the same as the true label from training labels, the True instances array is appended by 1, otherwise zero.

Then the ones of the array are summed and the accuracy computed with respect to the total testing Data size.

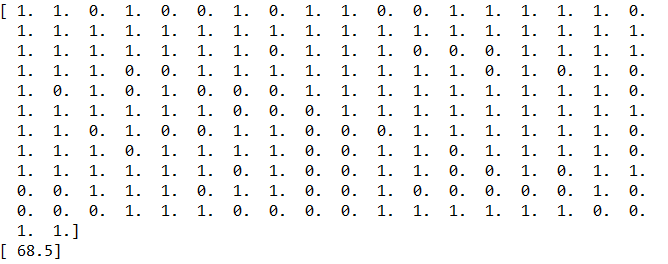


**Step4: Running PCA**

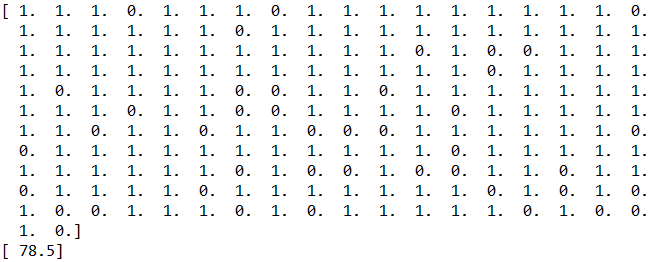
As asked in the Assignment, we ran the code on different Alpha values: 0.8, 0.85, 0.9, 0.95.

The results of the True instances matrix and the Accuracy **(last value)** came as follows

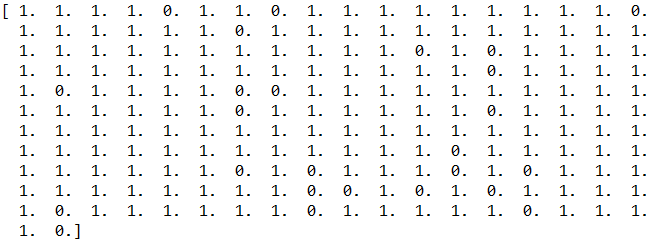
**Alpha = 0.8**



**Alpha = 0.85**

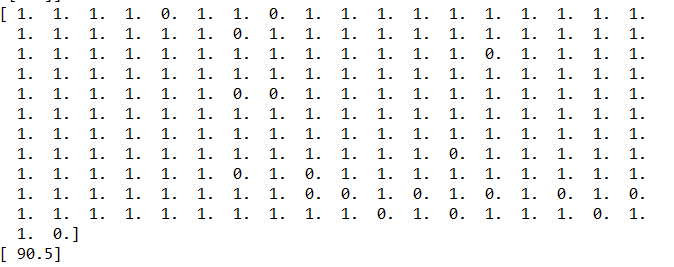


**Alpha = 0.9**

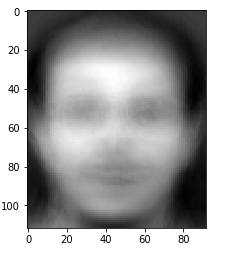




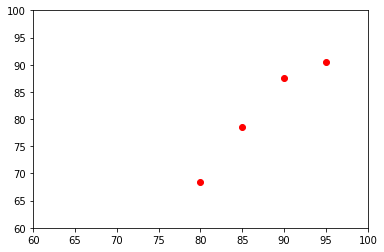
**Alpha = 0.95**



**Extra: How the “mean face” looks like**

****

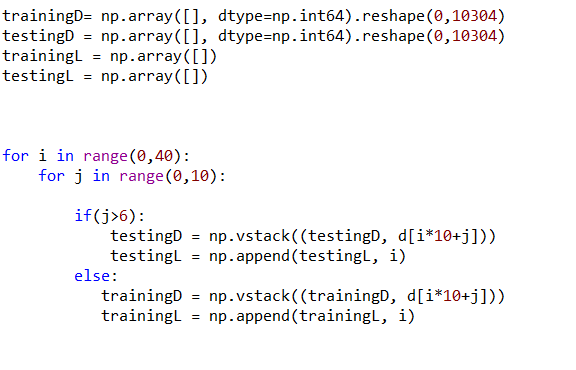
**Comparison between results of different alphas**

****

**Step5: (First Bonus) Running PCA with 70%-30% data split**

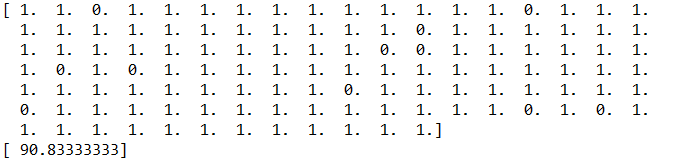
**a. Splitting the data**

We split the data by looping over the data matrix and appending the data and label below 7 to the training data/label and above to the testing data/label



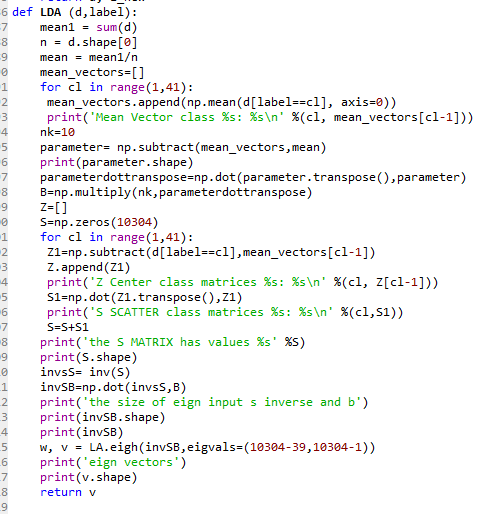
While the rest of the code is as it is.

**b. Running the code**

****

**Comment:** Naturally, because the Training data is larger, the accuracy is better.

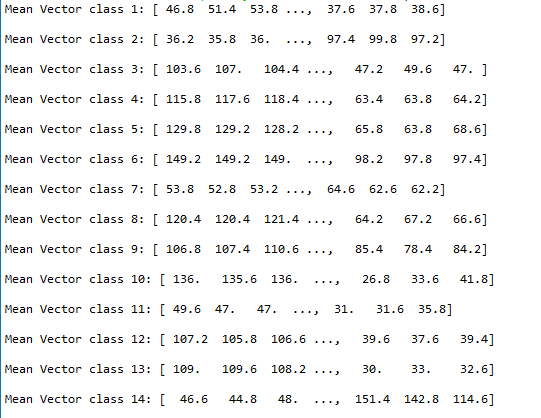
**Step6: LDA**



We start the LDA by

**1-** Computing the mean of all the data.

**2-** Computing the mean per each class we get 40 means in the mean vectors matrix



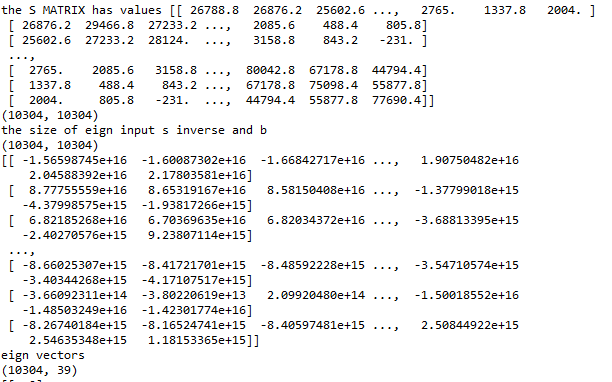
**3-** We subtract the mean of each class by the total mean of data calling that term parameter

**4-** We cross multiply the term parameter with it’s transpose and scaler multiply nk=10 the number of samples in each class to that term and generating the B between class scatter matrix

**5-** Calculate the Z the center class matrix where we subtract the data of the class from it’s mean

**6-** We calculate class scatter matrix for each of the 40 classes by having cross product of Z & it’s transpose.

**7-** We add all the class scatter matrix values to one S matrix



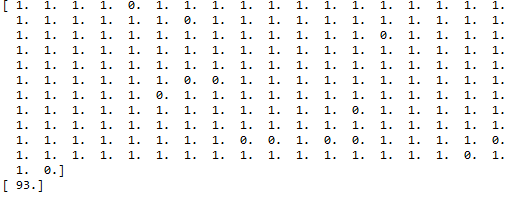
**8-** We get inverse of S matrix and cross it with the B matrix have the result in eign function

**9-** We extract the 39 domain eign vectors and return it .

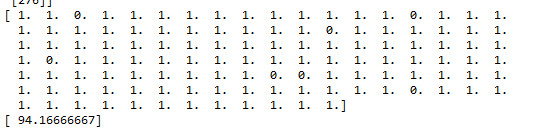
**10-** We project the data by cross product of the U with our training data and testing data sets

**11-** We report the accuracy by having a Nearest Neighbors algorithm.

**Note : our accuracy using 50% data splitting by LDA is 93**



**With splitting 70% our accuracy increased to 94.166**



**Step7: Conclusion**

**1-** LDA has better accuracy than PCA (remarkably 93 to 90.5 as the highest given alpha)

**2-** By splitting data 70%-30%, because then the Training data is larger, the accuracy gets better (but not too significantly in our case because the data number increases from 200 to 280).