Homework Assignment 5: PCA and Face Recognition

Due Monday, February 24, 2020 at 11:59 pm EST

Description and dataset instructions

In lectures, we discussed PCA for dimensionality reduction and its application for face recognition. The basic idea is that the first few Eigenvectors contains most of the variance of the data. For this problem set, you will implement the necessary elements to compute the Eigenfaces and apply different classification algorithms.

For this assignment, we are going to use the AT&T labs face dataset (Download zip file). There are ten different images of each of 40 distinct subjects. The size of each image is 112x92 pixels, with 256 grey levels per pixel. The images are organized in 40 directories (one for each subject), which have names of the form sX, where X indicates the subject number (between 1 and 40). In each of these directories, there are ten different images of that subject, which have names of the form Y.pgm, where Y is the image number for that subject (between 1 and 10).

Extract the dataset to the directory 'input/all' then write a function/script that randomly picks 8 images from each subject folder for training. Save these images under the directory 'input/train'. You should get a total of 320 images in this directory (you shouldn't use the same file names; otherwise you may get conflicts. Chose proper naming scheme). Those two images per subject that weren't selected for training should be saved under 'input/test/sX'. The directory 'input/test' should now contain 40 subfolders with 2 images in each subfolder.

What to submit

Download and unzip the ps6 folder: ps5.zip
Rename it to `ps5_LastName_FirstName` and add in your solutions:

`ps5_xxxx_LastName_FirstName/`

- input/ - input images, videos or other data supplied with the problem set
- output/ - directory containing output images and other files your code generates
- ps5.m - code for completing each part, esp. function calls; all functions themselves must be defined in individual function files with filename same as function name, as indicated
- *.m Matlab/Octave function files (one function per file), or any utility code
- ps5_LastName_FirstName_debugging.m – one m-file that has all of your codes from all the files you wrote for this assignment. It should be a concatenation of your main script and all of your functions in one file (simply copy all the codes and paste them in this file). In fact, this file in itself can be executed and you can regenerate all of your outputs using it.
- ps5_report.pdf - a PDF file with all output images and text responses

Zip it as `ps5_LastName_FirstName.zip`, and submit on Canvas.

Guidelines

1. Include all the required images in the report to avoid penalty.
2. Include all the textual responses, outputs and data structure values (if asked) in the report.
3. Make sure you submit the correct (and working) version of the code.
4. Include your name and ID on the report.
5. Comment your code appropriately.
6. Please avoid late submission. Late submission is not acceptable.
7. Plagiarism is prohibited as outlined in the Pitt Guidelines on Academic Integrity.

Questions

1. PCA analysis

In this part, you need to implement the PCA algorithm for face images. You need to write a code to compute the mean image solve for the eigenfaces.

a. Write the code to read all the images in the training directory. Reshape each image to be represented as one column vector. Construct a matrix T whose columns are the training images. The size of T should be 10304 x 320.

   Output: The gray level images showing the values of T as `ps5-1-a.png.` You can use this command in MATLAB: imshow(T,[]) to display matrix T as an image.
   `Hint: Due to the large difference in the dimensions of T, it's tricky to visualize T. You can zoom-in and add several screenshots from different regions in the image and include those screenshots in the report.`

b. Compute the average face vector m. The average face vector should be the 10304x1 mean vector computed across the column of T.

   Output: Resize m to 112x92 and display the resultant image (the mean face) as `ps5-1-b.png`

   Output (`textual response`):
   - Describe your results.

c. Find the centered data matrix, $A = T - m$, i.e., you subtract the mean vector from each column of T. Then define the data covariance matrix $C = AA^T$. Remember the dimension of C has to be 10304x10304.

   Output:
   - image of the covariance matrix as `ps6-1-c.png`

d. Use Matlab function '`eig`' to compute only the eigenvalue of $A^T A$. This will give you 320 values. Recall that each eigenvalue (λ) represent how much variance is retained by the corresponding eigenvector and that the first eigenvector captures the maximum variance. Now we are going to learn how to decide on how many eigenfaces are enough to represent the variance in our training set. To do so, we define the following

$$v(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{N} \lambda_i}, k = 1, 2, \cdots, N;\ and\ \lambda_m > \lambda_n\ \forall\ n > m$$

where N is the total number of eigenvalues (in our case, the total number of training images). $v(k)$ is the percentage of variance captured by the first k eigenvectors. Compute the values of $v(k)$ and determine the minimum number of eigenvector, K, needed to capture at least 95% of the variance in the training data (i.e., the minimum value of k such that $v(k) \geq 0.95$).
Hint: eig returns eigenvalues sorted in an ascending fashion. Before computing $v(k)$, you'll need to sort the eigenvalues in descending fashion, we need to keep those large values, not the residual ones!

Output: The plot of k vs $v(k)$ as `ps5-1-d.png`

Output (`textual response`):
- The number of eigenvectors, K, that capture 95% of the variance in the training data

e. Using the value of K you obtained from the previous question, retrieve the K dominant eigenvectors corresponding to the heights K eigenvalues. Save those dominant eigenvectors in a basis matrix U. Those vectors will be used as the basis for PCA dimensionality reduction (i.e., each image will be represented as a weighted sum of those vectors). Now, U defines the reduced eigenface space.
Hint: The MATLAB '`eigs`' function is very useful in this regard!
Output:
- image of the first 8 eigenfaces (you will need to resize the eigenvectors) in one figure as `ps5-1-e.png`

Output (`textual response`):
- The dimensions of matrix U
- Describe your results and comment on the eigenfaces you obtained.

2. Feature extraction for face recognition

For us to use different classification techniques, we need to extract some features. For face recognition, we are going to use the image representation in the reduced eigenface space as our feature vector. Any image can now be represented in the new space as $I = m + w_1 u_1 + w_2 u_2 + \cdots + w_K u_K$, where m is the mean vector from 1.b, u_i is the i^{th} column of the basis matrix U, and $w = [w_1, \cdots, w_K]^T$ is reduced representation of the image I in the reduced eigenface space. Compare the size of the image vector to the size of vector w. Definitely, there is a great deal of dimensionality reduction. As matrix multiplication, for one image I, w can be computed as $w = U^T(I - m)$, remember, I and m are now vectors, not $2D$ matrices.

a. Project all the images in the training folder in the new reduced eigenface space, i.e., find w for each image in the training folder. Construct a matrix W_training where each row in it corresponds to one reduced training image. W_training is the training features matrix.
Hint, keep track of which subject corresponds to which row in W_training, this will define your labels vector (you will need that later to train a classifier).

b. Project all the images in the testing folder in the new reduced eigenface space, i.e., find w for each image in the testing folder. Construct a matrix W_testing where each row in it corresponds to one reduced testing image. W_testing is the testing features matrix. Hint, keep track of which subject corresponds to which row, this will define your true_class vector (you will need that later to compute the accuracy of your classifier).

Output (`textual response`):
- The dimensions of W_training and W_testing

3. Face recognition

Next, you'll use the training features to train KNN and SVM classifiers and test the resultant classifier using the testing features. For this section, you can use available packages for K-NN and SVM.

a. Train a KNN classifier using W_training matrix and the associated labels vector. Test your classifier using samples in W_testing. Use K = 1, 3, 5, 7, 9, and 11 nearest neighbors. Compare the output of the classifier to the true_class and compute the classifier accuracy. Accuracy is defined as the number of correctly classified samples divided over the total number of testing samples.

Output (`textual response`):
- Table for your KNN classifier accuracy at the different values K listed above.
- comment on and discuss your results.

Use W_training matrix and the associated labels vector, train three SVM classifiers (you can use MATLAB `fitcsvm` and `predict`, but **not** `fitcecoc`). Each classifier must use a different kernel from others. Hence, use linear, 3rd order polynomial, and Gaussian rbf kernels, respectively. Since we have more than two classes, use the one vs all approach to build your multi-class classifiers. Compare the output of the classifier to the true_class and compute the accuracy of each classifier.

Output (`textual response`):
- table listing the accuracy of the SVM classifiers under different kernels
- comment on and discuss your results
- compare between the performance of KNN and SVM classifiers