EN.520.612.01.FA20 Machine Learning for Signal Processing
Laboratory 7 – Face Recognition using LDA

In this lab, we will employ LDA and SVM to identify the faces from different people.

We will use the ORL database, available on AT&T's web site, as in previous labs. This database contains photographs showing the faces of 40 people (s1, s2, . . . , s40). Each one of them was photographed 10 times. These photos are stored as images in levels of grey with 112x92 pixels. The data has been split in two parts: train and test. For each person, we use the first 9 photographs for training and the last photograph for test.

1. Load the training and the testing data (separately) and change each (d1 = 112) x (d2 = 92) photograph into a vector;

```
% You can create a function on the form:
% [trainingdata,testingdata]=loadImagesLab();
```

Classification using LDA

Apply LDA to the **training** set. Follow these steps:

2. Estimate the between class covariance and within class covariance. Then, use them to estimate the LDA matrix *V* that solves the following *generalized* Eigen problem

Sb.*v* = l.*Sw*.*v* (use the eigs Matlab function [V, D]=eigs(*SB* , *Sw* , L-1) to compute the eigenvectors), where *SB* contains the between class scatter and *Sw*, the within class scatter of the training images feature vectors (you can use PCA weights as features for training and testing). Remember that if you have *L* different classes, LDA will provide *L-1* non-zero eigenvalues.

```
%You can create several functions on the form:
% [TrainingPCA,TestingPCA]=PCAlab(trainingdata,testingdata,PCAdimension);
%
[V,D,TrainingLDA,TestingLDA]=LDAlab(TrainingPCA,TestingPCA,LDAdimension);
```

3. Check the identity corresponding to each photograph in the **testing** set by determining its LDA projection (with dimension 10, 20, 30 and 39) employing the LDA matrix obtaining with the training data and then comparing the distances of this projection with respect to all projections in the training data. For each test category find the closest category in the test. Report your accuracy

```
%You can create several functions on the form:
% [LDAresults]=distancesLab(TrainingLDA,TestingLDA);
% Also, you can generate figures representing the accuracy as a function
of the PCAdimension and LDAdimension
```

Optional: Classification using LDA+SVM (+5 extra points)

4. Using the LDA projections of the training faces, train a SVM model to identify the different people. Therefore, resulting model will include 40 classes. Then, test the model using the LDA projections of the testing faces. Use the Matlab function or LIBSVM library to do the binary classification. Examples can be found in https://www.mathworks.com/help/stats/support-vector-machine-classification.html for MATLAB. Play with different C values and kernel functions and see how they influence the result. Report your best accuracy and settings include dimension, C value, kernel function you used.

```
% [SVMresults]=SVMlab(TrainingLDA,TestingLDA);
% You can generate figures representing the accuracy as a function of the
PCAdimension and LDAdimension and C
```

Provide a report in *pdf* format explaining your results (you can provide a pdf of this script including graphics and results).

NOTES:

- The suggested functions are just a guide. You can create them or not. Also, you can add or remove input/output parameters as needed.
- Between class scatter:

$$S_B = \sum_{k=1}^{K} N_k (m_k - m)(m_k - m)^T$$

where *K* is the total number of classes, N_k is the number of observations on class *k*, m_k is the mean of vectors of class *k* and m is the mean of all the vectors (global mean).

- Within-class scatter:

$$S_w = \sum_{k=1}^{K} \sum_{i=1}^{N_k} (x_{ik} - m_k)(x_{ik} - m_k)^T$$

where x_{ik} is the i^{th} observation of class *k*.

- PCA projection is recommended to obtain the features to be used to train the LDA matrix in order to avoid singularity problems. If PCA is not used over the original data, the number of features per observation (10304) will be much larger than the number of observations (360) and the covariance matrices will not have full rank and will not be invertible.

Submission instructions:

Deliverables:

- All of your MATLAB (.m) files (All .m files needed to run your code)
- A report in PDF format (Include everything according to the instructions above)

Compress the deliverables into one zip file and name it 'lab6_<your JHED id>'. Submit the zip file on Canvas.

The classification algorithm based on LDA and cosine scoring is described as follow

- Normalizing the length of all I-vectors x (all directories).

$$w = \frac{x}{\|x\|}$$

- **LDA training**

  - Estimate the between class covariance

$$S_b = \sum_{l=1}^{L} n_l (w_l - \bar{w})(w_l - \bar{w})^t$$

  - Estimate the within class covariance

$$S_w = \sum_{l=1}^{L} \sum_{i=1}^{n_l} (w_i^l - w_l)(w_i^l - w_l)^t$$

  where

$\bar{w}$: the mean of the all I-vectors

$w_l = \frac{1}{n_l} \sum_{i=1}^{n_l} w_i^l$ mean for language class l

n_l = number of I-vectors for each language class l

L = total number of language classes

The goal is the estimate LDA matrix V that solves the following Eigen problem $S_b.v = \lambda.S_w.v$ (Use the eigs Matlab function [V, D]=eigs(S_b ,S_w, L-1) to compute the eigenvector). If you have L different classes, LDA will provide L-1 non-zero eigenvalues.

- **Classifier training**
  - Project the i-vectors with LDA matrix V then length normalized the obtained projected I-vectors

$$w' = \frac{V^t w}{\|V^t w\|}$$

  - For each class l compute the mean and normalized its length

$$m_i = \frac{\frac{1}{n_l} \sum_{j=1}^{n_l} w'}{\left\| \frac{1}{n_l} \sum_{j=1}^{n_l} w' \right\|}$$

- **Classifier testing**

  - Project the test i-vectors in both dev and eval directories by LDA matrix V

    $$w'_{test} = \frac{V^t w_{test}}{\left\| V^t w_{test} \right\|}$$

  - Compute the dot product of the test i-vector with the normalized mean of each class m_i

    $$score_i = w'_{test} * m_i$$

  - Select the language class number i that corresponds to do highest score

  - Compute both recognition accuracy rates for dev and eval data

    $$Acc_{dev} = \frac{\text{number of correct classified I-vector in dev directory}}{\text{total number of I-vector in dev directory}}$$

    $$Acc_{eval} = \frac{\text{number of correct classified I-vector in eval directory}}{\text{total number of I-vector in eval directory}}$$