Due date: November 14 at 23h59.

Problem 1: Overcomplete sparse presentation

In this problem, the goal is to build a language recognition system based on I-vector representation. The I-vector presentation approach consists of representing speech segments in single space. This space will felicitate the comparison between all the recordings. In the directory /Problem1/data/, you will find three directories Train, Dev and Eval. These directories correspond respectively to the training data that will be us to train your classifier, development data to tune your classifier and evaluation data to evaluate the different models. In each directory you will find 24 files that correspond to 24 different language classes. Each line of these 24 files corresponds to the I-vector x of dimension 600 and represents a speech recording. The goal of this problem is to build two types of classifier based on Linear Discriminant Analysis (LDA) and sparse dictionary representation.

I) Write the code that applies LDA and cosine scoring to the I-vectors provided in /Problem1/data/. 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 function 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

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

  - 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

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

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

II) In the question the goal is to write the code that classifier the I-vector x based on overcomplete space representation (see similar work for face recognition http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4483511&tag=1). If you load each of the training data file in matrix D_c (the column of this matrix correspond to the I-vector which is one line in the training file), you can build your dictionary for the overcomplete sparse presentation by concatenating all the 24 matrices in single large matrix $\mathrm{A}=[\mathrm{D}_1, \mathrm{D}_2, ..., \mathrm{D}_{24}]$.
For each test example in Dev and Eval directories uses Lasso algorithm to estimate the new weights in the Dictionary space.

$$\min\left\{\left\|x - A\alpha_i\right\|^2 - \lambda\left\|\alpha\right\|_1\right\}$$

where $\alpha^t = \left[\alpha_1^t, \alpha_2^t, ..., \alpha_{24}^t\right]$ and each α_c^t is a vector of weights corresponding to the language class c.

The classification in the overcomplete sparse presentation is done based on minimizing the reconstruction of the I-vector x based on each D_c matrix and its weights α_c. This task consists of finding the language class that produces the small reconstruction error for each I-vector $\underset{c}{\operatorname{argmin}}\left\{\left\|x - D_c\alpha_c\right\|^2\right\}$ on both directories for Dev and Eval then compute the recognition accuracy rates?

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

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

III) In this question, please redo the same process as question (II) after projecting first all the I-vectors by LDA, which was trained in question (I). Evaluate again the recognition accuracy rates for both Dev and Eval directories? If both accuracies are better than question (II), explain why?

IV) To speed up the process, train a code book using k-means for each language class with k=55. Each language will be only represented by 55 centroids. The classes that have already 55 I-vectors take all of them (do not apply kmeans). For kmeans implementation used the function already predefines in python package.

Apply the overcomplete space representation classification again. Similar to question III use the after projecting first all of them by LDA

V) Apply Gaussian classifier as it will be seen in the class for this task with the assumption of equal prior for all the language classes and shared covariance between all the classes. In this part use the **original I-vector of dimension 600**. Compare your results with previous ones.

For reference look to
http://www.fit.vutbr.cz/research/groups/speech/publi/2011/martinez_interspeech2011_291.pdf

Submission Instructions:

- Submit all your code and a pdf report as a single zip file.

- The report includes theoretical answers and accuracies for all questions.

- Do not include the Data in submission folder (for easy uploading and downloading).

Solutions should be uploaded to CANVAS.

If you use MATLAB, use these templates. If you use Python, use the Jupyter notebook template for all problems in google collab in the following link-

https://colab.research.google.com/drive/1RImIu1vGJKWxppGE_lq_GB6rE37aoe7M?usp=sharing

Instructions for MATLAB:

You will submit a zip file containing all the required files specified below. Title your zip file as "HW4_yourJHID.zip", where 'yourJHID' is your JHED ID.

Instructions for Python:

You will have to use Jupyter Notebook for Python. We provided the template in Google Colab. Instructions are given in the link. You have to submit the specific deliverables mentioned below. Put them inside "results" folder inside the designated problem folder. You have to submit the "ipynb" file and "HW4_yourJHID.zip" zip file containing the required results or reports. In the text comments on Canvas, put the link to your Google colab notebook with proper permissions.