**Homework 1 – Intro, functions, prepare data**

We want to implement a classifier able to predict which digit is pronounced in a short audio excerpt.

To achieve this, we used the following libraries:

Se aumenti n-coeff esce più accurato?

import numpy as np import sklearn.svm

import librosa import IPython.display as ipd

import os import scipy as sp

import matplotlib.pyplot as plt

In the first place we have collected our data from the folders of the original dataset in the vector train\_root:

! git clone <https://github.com/Jakobovski/free-spoken-digit-dataset.git>

train\_root = ('free-spoken-digit-dataset/recordings')

The dataset consists of 3000 recordings of spoken digits in wav files at 8kHz.

The recordings are trimmed so that they have near minimal silence at the beginnings and ends

Furthermore files are named in the following format:

{digitLabel}\_{speakerName}\_{index}.wav Example: 7\_jackson\_32.wav

In the second place we have thought which set of features can be useful for our task and that can guarantee best classification results.

**Functions**

We have come to the conclusion that we can use as main feature the mel-frequency cepstrum coefficients. In fact MFCCs can be used as an excellent feature vector for representing the human voice and musical signals.

In particular we know that the MFCC parametrization of speech has been proven to be beneficial for speech recognition.

The MFCCs are based on a subband filtering by means of a series of triangular filters whose center frequencies are equally spaced according to the mel scale.

In order to compute the filters we have defined the function that takes in input the audio array, the frequency rate and the number of MFCCs:

def compute\_mfcc(audio, fs, n\_mfcc):

In the code we have computed the spectrogram of the audio signal , this by taking the absolute value of the Short Time Fourier Transform:

X = np.abs(librosa.stft(audio,

                        window='hamming',

                        n\_fft = 512, #???????????????????????????????

                        hop\_length= 256,

                       ))

After that, we have calculated the weights of the mel filters with the filters.mel of librosa:

mel = librosa.filters.mel(

sr=fs,

        n\_fft = 512,

        n\_mels = 40,

        fmin = 133.33,

        fmax = 4000

    )

we have applied the filters to spectrogram :

melspectrogram = np.dot(mel, X)

and taken the logarithm:

log\_melspectrogram = np.log10(melspectrogram + 1e-16)

At the end we have computed the DCT to log melspectrogram to obtain the coefficients:

    mfcc = sp.fftpack.dct(log\_melspectrogram, axis = 0, norm='ortho')[1:n\_mfcc+ 1]

**Prepare data**

Now we must organize and prepare our data by splitting our dataset in training set and testing set.

The test set officially consists of the first 10% of the recordings. Recordings numbered 0-4 (inclusive) are in the test set and 5-49 are in the training set.

In this way we can build a dataset that is balanced and has an equal number of samples for each class. Furthermore, training set and testing set are as different as possible, which is a necessary condition in order to have a more efficient classifier.

We defined our classes, one for each digit, and the number of MFCCs:

classes = [0,1,2,3,4,5,6,7,8,9]

n\_mfcc = 13

Empty dictionary, where each subclass is the digit:

dict\_train\_features = {0: [], 1: [], 2: [],3: [],4: [],5: [],6: [],7: [],8: [],9:[]}

dict\_test\_features = {0: [], 1: [], 2: [],3: [],4: [],5: [],6: [],7: [],8: [],9:[]}

We have allocated in the class\_train\_files variable all the train\_root files:

class\_train\_files = [f for f in os.listdir(train\_root) if f.endswith('.wav')]

After that, for every file contained in class\_train\_files we have splitted the string of the name of each file in a vector containing 3 strings: digit, speaker, number of recording of the speaker.

We computed this by using a temporal variable that will be rewritten at every ieration:

  tmp = class\_train\_files[i];

  tmp = (tmp.split('.'))[0].split('\_');

With librosa at every iteration we obtained the audio array of the current .waw file at the original frequency.

We calculated the MFCC’s and compute the mean value for every frequency band along the axix=1 (columns):

  audio, fs = librosa.load(os.path.join(train\_root,class\_train\_files[i]), sr= None)

  mfcc = compute\_mfcc(audio, fs, n\_mfcc)

  tmp\_features = np.mean(mfcc, axis=1);

After that, as said before, if the recording is numbered from 5 to 49 we have put the n-th audio in the dict train, by placing it in the correct digit class of our dictionary:

int(tmp[0]

  if int(tmp[2]) > 4:

    dict\_train\_features[int(tmp[0])].append(tmp\_features)

Otherwise, if the recording is numbered from 0 to 4 we have put it in the dict test feature:

  else:

    dict\_test\_features[int(tmp[0])].append(tmp\_features)

We named in this way this lengths, that will be useful in the next lines of the code:

Len\_train = len(dict\_train\_features[0])

Len\_test = len(dict\_test\_features[0])

Len\_digit = 10