ALESSANDRO SCALAMBRINO 923216

AUDIO ANALYSIS AND CLASSIFICATION PROJECT

Introduction

What’s the ‘goal’ of this project? We want to classify different kinds of sounds. The principal issue lies in the definition of "sound type" itself, because a particular sound, is classified using too many different aspects which are mostly subjective.

Sounds types have no strict definitions, they can change using one point of view or another, as they are a product of multiple factors.

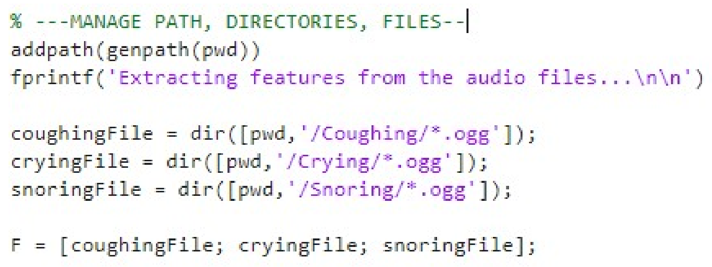
For this project, I chose human nature sounds to show how it is possible to use audio analysis to perform complex analyses, which may have applications in the medical and security fields.

The first step is reducing the information we want to analyze using only useful informations. We may be be interested in different features like how the spectrum is distributed or how the sound is loud or how it changes his loudness.

1)DIRECTORY AND FILES MANAGEMENT

we have three different categories of sounds. I decided to choose sounds produced by human beings: coughing sounds, crying sounds, snoring sounds.

Each folder contains 40 samples of sounds, in the following code we tell matlab the relative path and directories it should use.



```
% ---MANAGE PATH, DIRECTORIES, FILES--
addpath(genpath(pwd))
fprintf('Extracting features from the audio files...\n\n')

coughingFile = dir([pwd,'/Coughing/*.ogg']);
cryingFile = dir([pwd,'/Crying/*.ogg']);
snoringFile = dir([pwd,'/Snoring/*.ogg']);

F = [coughingFile; cryingFile; snoringFile];
```

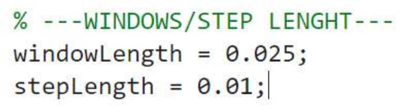
2)FEATURES EXTRACTION

Short-term feature extraction

The audio signal is divided in windows. For each window it's calculated a vector of features, some based on time signal representation and some others on frequency representation.

From this operation it's obtained a vector of N elements, depending on the duration of the audio considered, his sample frequency, the window size chosen and the how much the windows are overlapped.

In this case I chose the following values (25 ms), because they gave me the best performance



```
% ---WINDOWS/STEP LENGHT---
windowLength = 0.025;
stepLength = 0.01;
```

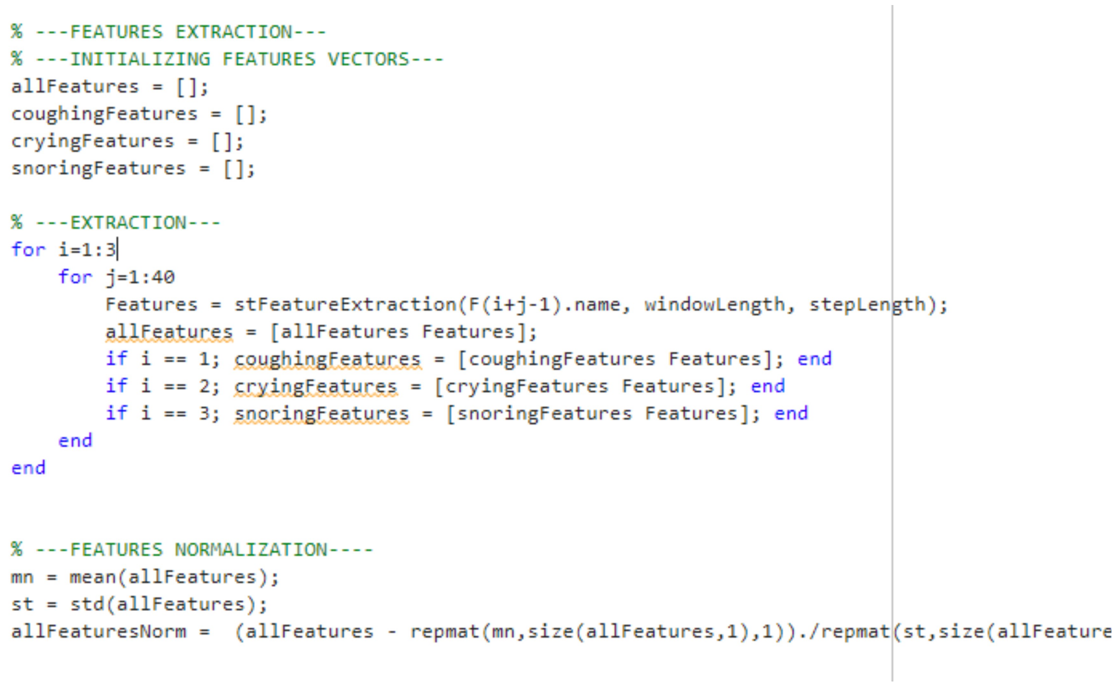
Features in time domain (energy and zero crossing rate)

The Feature in time domain to examinee are the Energy (magnitude) of the signal and the Zero-Crossing Rate, the frequency at which the signal changes from positive to negative or back.

Features in frequency domain (spectral centroid and spread, spectral rolloff and mel-frequency cepstral coefficients)

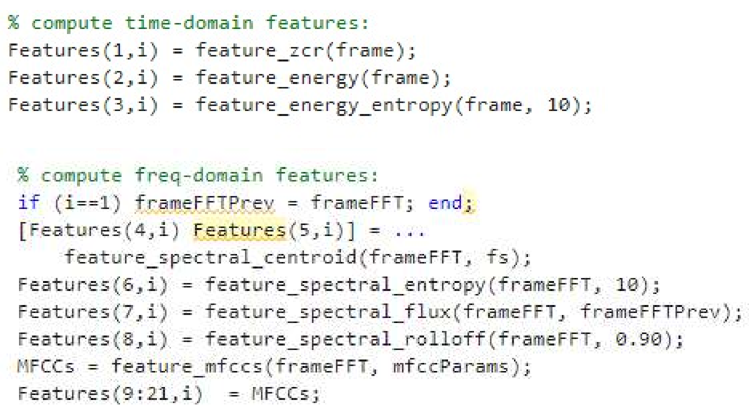
The spectrum can be described using simply two features: the centroid which is the main point of the spectrum distribution and the spread which is the standard deviation of a spectrum distribution.
The spectral rolloff calculates the amount of energy cumulated until a certain point in the frequency.
Mel-Frequency Cepstral Coefficients is a feature which represents the spectrum bands according to the mel-scale, that is an isophonic (mostly subjective) coefficient.

<u>Feature extraction code</u>:



```
% ---FEATURES EXTRACTION---
% ---INITIALIZING FEATURES VECTORS---
allFeatures = [];
coughingFeatures = [];
cryingFeatures = [];
snoringFeatures = [];

% ---EXTRACTION---
for i=1:3
    for j=1:40
        Features = stFeatureExtraction(F(i+j-1).name, windowLength, stepLength);
        allFeatures = [allFeatures Features];
        if i == 1; coughingFeatures = [coughingFeatures Features]; end
        if i == 2; cryingFeatures = [cryingFeatures Features]; end
        if i == 3; snoringFeatures = [snoringFeatures Features]; end
    end
end


% ---FEATURES NORMALIZATION----
mn = mean(allFeatures);
st = std(allFeatures);
allFeaturesNorm =  (allFeatures - repmat(mn,size(allFeatures,1),1))./repmat(st,size(allFeature
```

<u>the core of features extraction is the stFeatureExtracion() function</u>:



```
% compute time-domain features:
Features(1,i) = feature_zcr(frame);
Features(2,i) = feature_energy(frame);
Features(3,i) = feature_energy_entropy(frame, 10);

 % compute freq-domain features:
 if (i==1) frameFFTPrev = frameFFT; end;
 [Features(4,i) Features(5,i)] = ...
     feature_spectral_centroid(frameFFT, fs);
 Features(6,i) = feature_spectral_entropy(frameFFT, 10);
 Features(7,i) = feature_spectral_flux(frameFFT, frameFFTPrev);
 Features(8,i) = feature_spectral_rolloff(frameFFT, 0.90);
 MFCCs = feature_mfccs(frameFFT, mfccParams);
 Features(9:21,i)  = MFCCs;
```

<u>stFeatureExtracion() extracts features (time and frequencies domain) using functions provided during the course.</u>

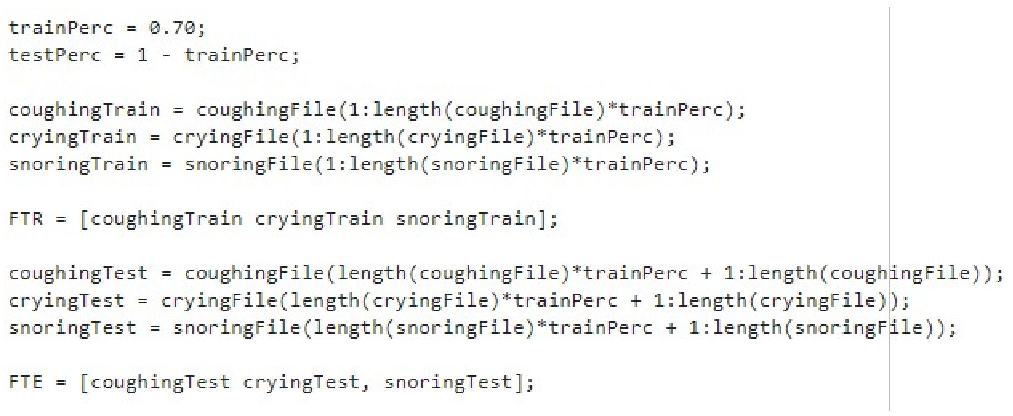
PCA

The results are combined into a single matrix (allFeatures) and normalized (allFeaturesNorm) on which we calculate the pca. Principal component analysis (PCA) is a technique used to identify a smaller number of uncorrelated variables known as principal components from a larger data set.

3)TRAIN/TEST SETS

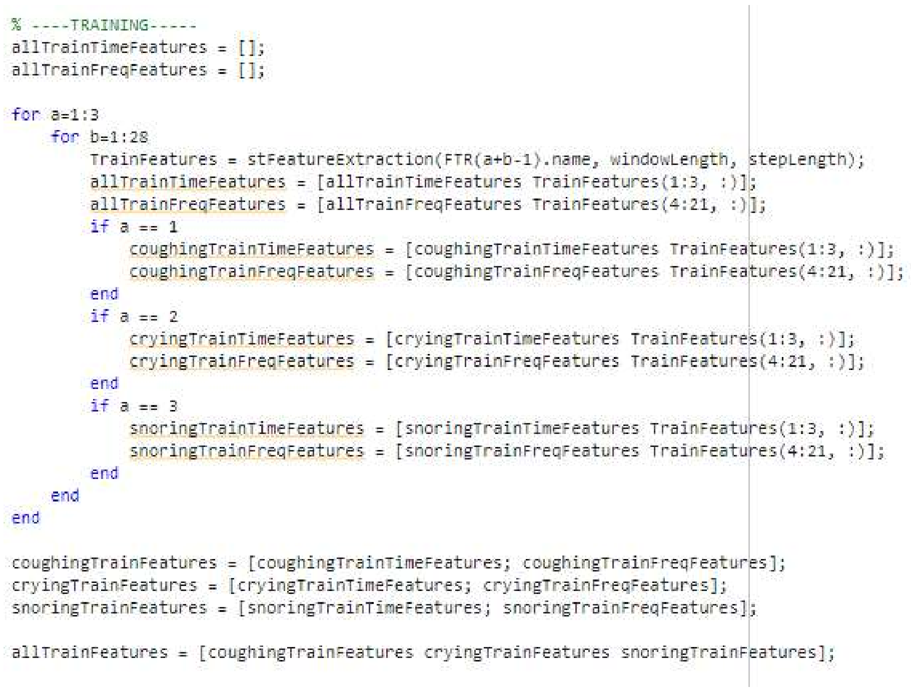
The audio samples are divided into two sets and normalized: train (70% of the data) and test (30% of the data). For each set we extract the features of time/frequency/time&frequency. The data are normalized and labels are applied. At this point we are ready to classify the data.

<u>Preparation of the train/test datasets:</u>



```
trainPerc = 0.70;
testPerc = 1 - trainPerc;

coughingTrain = coughingFile(1:length(coughingFile)*trainPerc);
cryingTrain = cryingFile(1:length(cryingFile)*trainPerc);
snoringTrain = snoringFile(1:length(snoringFile)*trainPerc);

FTR = [coughingTrain cryingTrain snoringTrain];

coughingTest = coughingFile(length(coughingFile)*trainPerc + 1:length(coughingFile));
cryingTest = cryingFile(length(cryingFile)*trainPerc + 1:length(cryingFile));
snoringTest = snoringFile(length(snoringFile)*trainPerc + 1:length(snoringFile));

FTE = [coughingTest cryingTest, snoringTest];
```

<u>Training code example:</u>



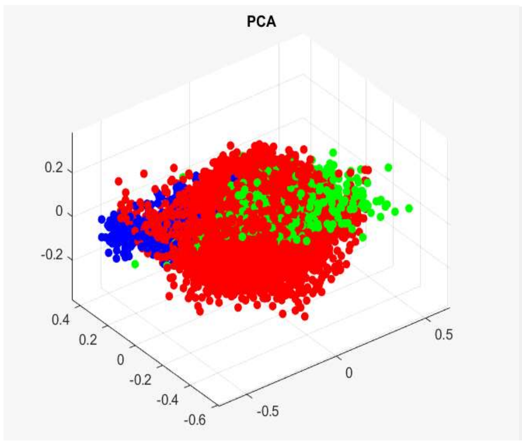
```
% ----TRAINING-----
allTrainTimeFeatures = [];
allTrainFreqFeatures = [];

for a=1:3
    for b=1:28
        TrainFeatures = stFeatureExtraction(FTR(a+b-1).name, windowLength, stepLength);
        allTrainTimeFeatures = [allTrainTimeFeatures TrainFeatures(1:3, :)];
        allTrainFreqFeatures = [allTrainFreqFeatures TrainFeatures(4:21, :)];
        if a == 1
            coughingTrainTimeFeatures = [coughingTrainTimeFeatures TrainFeatures(1:3, :)];
            coughingTrainFreqFeatures = [coughingTrainFreqFeatures TrainFeatures(4:21, :)];
        end
        if a == 2
            cryingTrainTimeFeatures = [cryingTrainTimeFeatures TrainFeatures(1:3, :)];
            cryingTrainFreqFeatures = [cryingTrainFreqFeatures TrainFeatures(4:21, :)];
        end
        if a == 3
            snoringTrainTimeFeatures = [snoringTrainTimeFeatures TrainFeatures(1:3, :)];
            snoringTrainFreqFeatures = [snoringTrainFreqFeatures TrainFeatures(4:21, :)];
        end
    end
end

coughingTrainFeatures = [coughingTrainTimeFeatures; coughingTrainFreqFeatures];
cryingTrainFeatures = [cryingTrainTimeFeatures; cryingTrainFreqFeatures];
snoringTrainFeatures = [snoringTrainTimeFeatures; snoringTrainFreqFeatures];

allTrainFeatures = [coughingTrainFeatures cryingTrainFeatures snoringTrainFeatures];
```

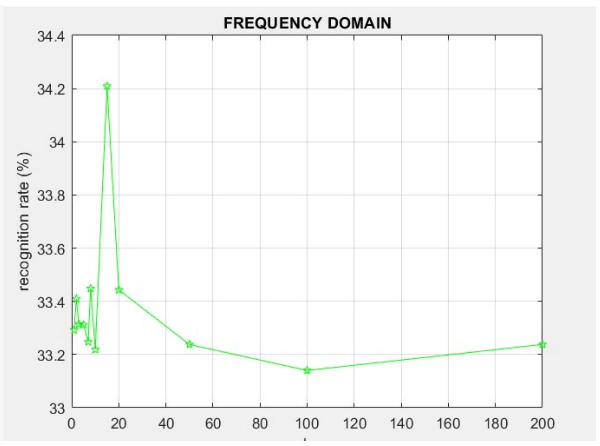
3)CLASSIFICATION

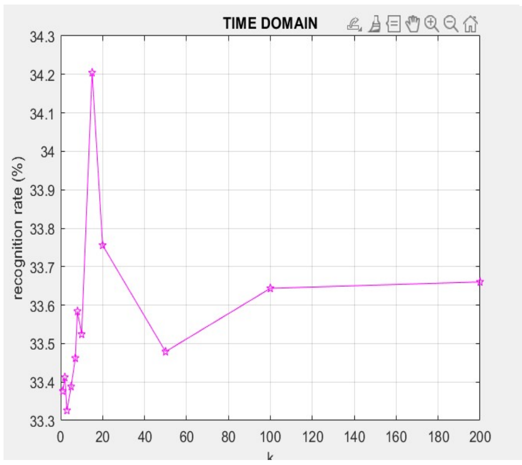
Nearest Neighbor Classifier

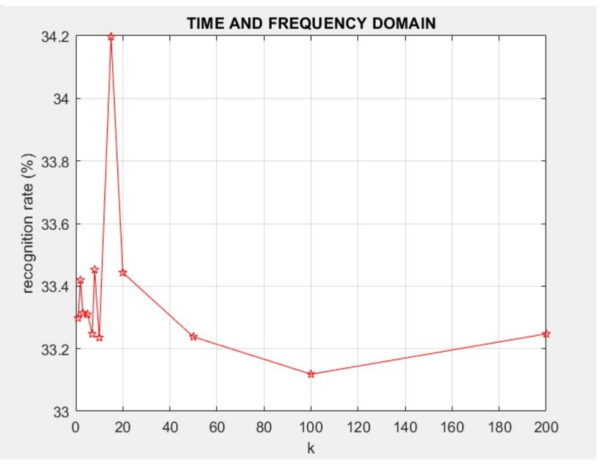
The classifier works using the kNN algorithm. When a new test is proposed it calculates the euclidean distance between the k nearest neighbours according to the euclidean distance in a multi-dimensional space, where the value for every dimension is given by the value of every single feature used in the predictor.
When the k neighbour are identified it takes place what is called a Majority Vote: the new test will be labeled according to the label which appears most between the k neighbours.
Trying with different values for k, it's possible to optimize the results, in this case I run the script with different intervals of k [1, 2, 3, 5, 7, 8, 10, 15, 20, 50, 100, 200].

4)RESULTS









The maximum recognition rate is 34.1972745280542 and it is achieved with a number of 15 nearest neighbors. We can see that in this case, increasing the number of knn neighbors to perform the calculation is counterproductive.The low recognition rate is due to the very similar and hardly distinguishable nature of the chosen sounds.