Data Analytics for the Smart Society

Lab Exercise 2: Speaker Identification

Emilie Krutnes Engen
100356077

Master in Big Data Analytics
Carlos III University of Madrid

Universidad
Carlos III de Madrid

1 Introduction

In this exercise we implement a Speaker Identification System based on Gaussian Mixture Models. The aim of the system is to predict the speaker ID given a speaker utterance from an unknown speaker. We first train a Gaussian Mixture Model for each speaker using a training set. Then the system is tested in both clean and noisy conditions. The speaker is identified by selecting the speaker ID which obtain the maximum log likelihood value. Finally the predicted speaker ID's are compared to the actual ID's in each of the test sets and the accuracy in both test sets are calculated.

2 Implementation

2.1 Training the Model

The first development phase consists of training the models. We first load the training data for each speaker by running the Matlab function *load_train_data()* for the list of files given in *list_train.txt* with a specific speaker ID.

To build the models of the different speakers in the system we first extract the acoustic features from the training data using the Mel-Frequency Cepstral Coefficients (MFCC). To perform the feature extraction using the MFCC parameters we use the Matlab function *melfcc()* from the RASTAMAT toolbox (**?**). The Matlab code for loading the training data and feature extraction for each training set, corresponding to a specific speaker is presented below.

```
num_spk = 16;                % Number of speakers
MFCCs = cell(1,num_spk);     % Cell array with features for all speakers

for i=1:num_spk
    % Create train set for each speaker
    Xtrain = load_train_data('list_train.txt', i);
    % Extract features for each speaker using MFCC
    sr = 16000;
    [MFCC,~] = melfcc(Xtrain, sr, 'numcep', 20,'wintime', 0.02,'hoptime', 0.01);
    % Transpose matrix
    MFCC = MFCC.';
    % Store features for speaker i
    MFCCs{1,i} = MFCC;
end
```

The window size, *wintime*, frame period, *hoptime*, and the number of cepstral components, *numcep*, are set to 20 ms, 10 ms and 20, respectively. The sampling frequency of the audio files is 16000 Hz and hence given by $sr = 16000$. The output features $MFCC$ for each speaker is stored in a cell array.

We proceed by building the speaker GMM models by applying the Matlab function *fitgmdist*, which implements the EM algorithm. A set of GMM models, one for each of the speakers, with a number of Gaussian components N_GMM are fitted to the features extracted from the training data corresponding to the specific speaker. The Matlab code for building the GMM models are presented below.

```
% Build GMM models with N_GMM components fitted to training data
N_GMM = 8;
GMModels = cell(num_spk,1);
for j=1:num_spk
    options = statset('MaxIter', 500);
    GMModels{j} = fitgmdist(cell2mat(MFCCs(1,j)),N_GMM,'Options',options,...
        'Replicates', 3, 'CovarianceType', 'diagonal');
end
```

2.2 Testing the Model

The implemented system is now tested using the test files listed in *list_test1.txt* and *list_test2.txt*, corresponding to test files in clean and noisy conditions. Each file is loaded one by one by running the Matlab function *load_test_data()*. The input is the text file with the listed files and a number i representing the specific file. Both the data and the label for each file is returned. The Matlab function *load_test_data()* is presented below.

```
function [x,y] = load_test_data(list_name, i)

% Read the list of the test speech files
fid = fopen(list_name);

if fid < 0
    fprintf('File %s does not exist\n', list_name);
    exit
end
info_speech = textscan(fid, '%s%f');
clase = int16(info_speech{2});    % speaker id of each file
fclose(fid);

% Extract the test speech samples of the specific
% "speaker_id" speaker
x = [];
y = [];
fname = info_speech{1}{i};
[aux fsaux] = audioread(fname);
id = clase(i);
x = [x; aux];
y = [y; id];
```

Then for each file we extract the features, applying the *melfcc* function previously described. After extracting the features for a given file, these features are compared to each of the models obtained from the training phase and the log-likelihood values are calculated for all models. For each of the files in the test set, the speaker identification is given by the speaker whose model obtain the maximum log-likelihood value. The Matlab code for loading the test files, extracting the features and predicting the speaker ID's for Test 1 are presented below. Test 1 is testing the system in clean conditions. The same approach is conducted for Test 2, which tests the system in noisy conditions.

```
% Test 1
fid = fopen('list_test1.txt');
info_speech_1 = textscan(fid, '%s%f');
numfich_1 = length(info_speech_1{1}); % total number of test files
fclose(fid);

% Create empty arrays for predicted and actual labels
pred_1 = [];
test_1_id_list = [];

for n=1:numfich_1
    % Load test file
    [Xtest_1,ytest_1] = load_test_data('list_test1.txt',n);
    % Extract features
    [MFCCt1,~] = melfcc(Xtest_1, sr, 'numcep', 20,'wintime', 0.02,'hoptime',...
    0.01);
    % Transpose matrix
    MFCCt1 = MFCCt1.';

    % Get log likelihood values
    p1 = [];
    for j=1:num_spk
        p = sum(log(pdf(GMModels{j}, MFCCt1)));
        p1 = [p1;p];
    end

    % Identify speaker: Get index given by max loglikelihood value
    [~, pred_id_1] = max(p1);
    pred_1 = [pred_1;pred_id_1];
    test_1_id_list = [test_1_id_list;ytest_1];
end
pred_1 = int16(pred_1); % convert to int array
```

We can now calculate the accuracy of the model in both clean and noisy conditions by computing the confusion matrix for both tests. From the confusion matrix the elements correctly classified are given on the diagonal. Hence, the accuracy of the system is given by the sum of the diagonal over sum of the entire matrix. The corresponding Matlab code for Test 1 is presented below.

```
% Test 1: Confusion matrix
confMatrix_test_1 = confusionmat(test_1_id_list,pred_1);

% Accuracy: compute accuracy score based on confusion matrix
accuracy_test_1 = sum(diag(confMatrix_test_1))/sum(confMatrix_test_1(:));
```

3 Improving the System

To investigate further improvements of the baseline system, some additional work is conducted. Initially the number of Gaussian components is set to 8. To investigate the effect of the number of components, the system is also tested for N_GMM ranging from 8 to 32. Other models is also explored. First a system extracting the features by applying the Linear Predictive Coding (LPC) coefficients, as oppose to MFCC is evaluated, and finally we test a system applying both MFCC and LPC.

4 Results

In Figure 1 the accuracy of the different models are plotted with respect to the number of Gaussian components. From the accuracy plot we observe that the three models all show a high accuracy, tested in clean conditions. The results are relatively stable with respect to the number of Gaussians used. When tested in noisy conditions we observe that the accuracy is noticeably reduced in for all models. From the plot we also observe that the accuracy is less stable when tested in noisy conditions.

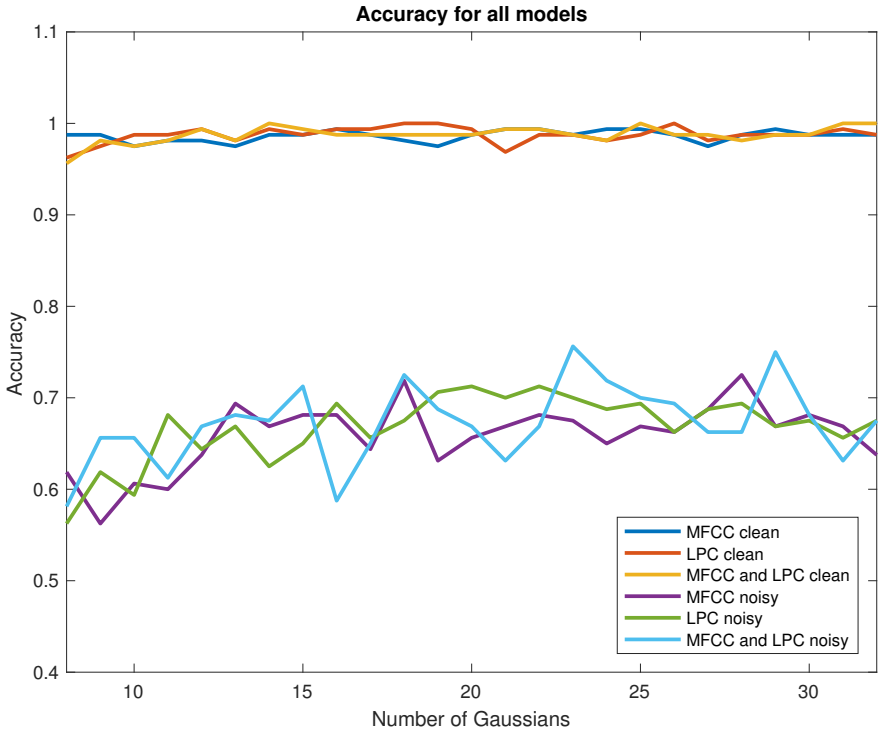

Figure 1: Accuracy in all models for different number of Gaussian components

When looking at the performance for the tests in clean conditions in Figure 2, we observe a slightly positive trend for the model applying both MFCC and LPC with respect to the number of Gaussian components. We further observe that LPC alone give a slightly higher variability in the accuracy. However all models have an accuracy above 95 % regardless of the number of Gaussian components used.

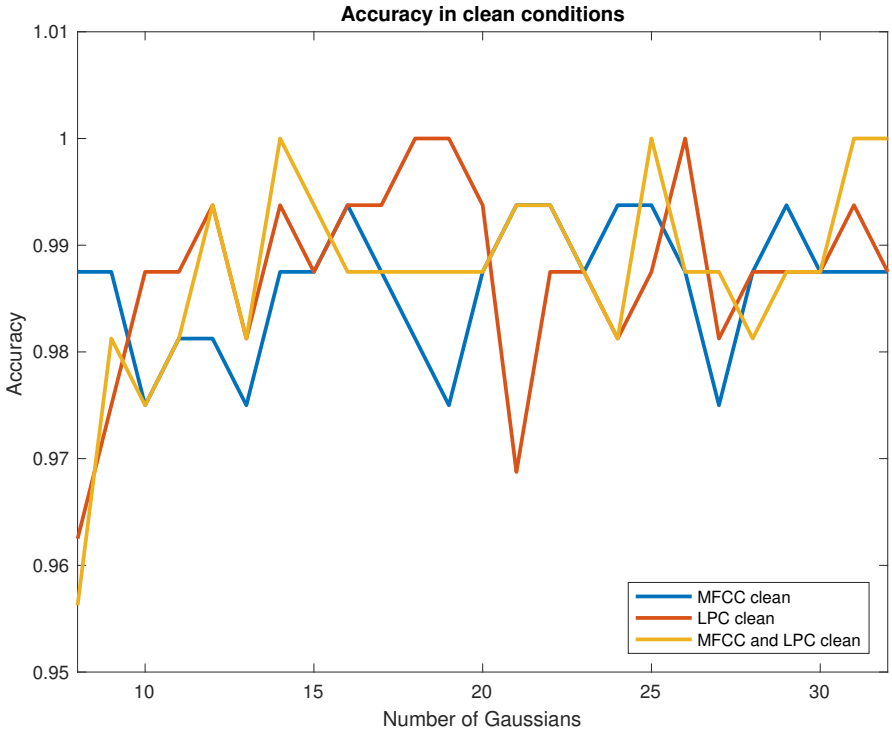

Figure 2: Accuracy in all models tested in clean conditions for different number of Gaussian components

In Figure 3 the accuracy of models tested in noisy conditions are presented. From the plots we observe that the maximum accuracy is 75 % for the combination of MFCC and LPC, given by using 24 Gaussian components. Here the combination of MFCC and LPC give a slightly higher variability in the accuracy with respect to the number of Gaussians.

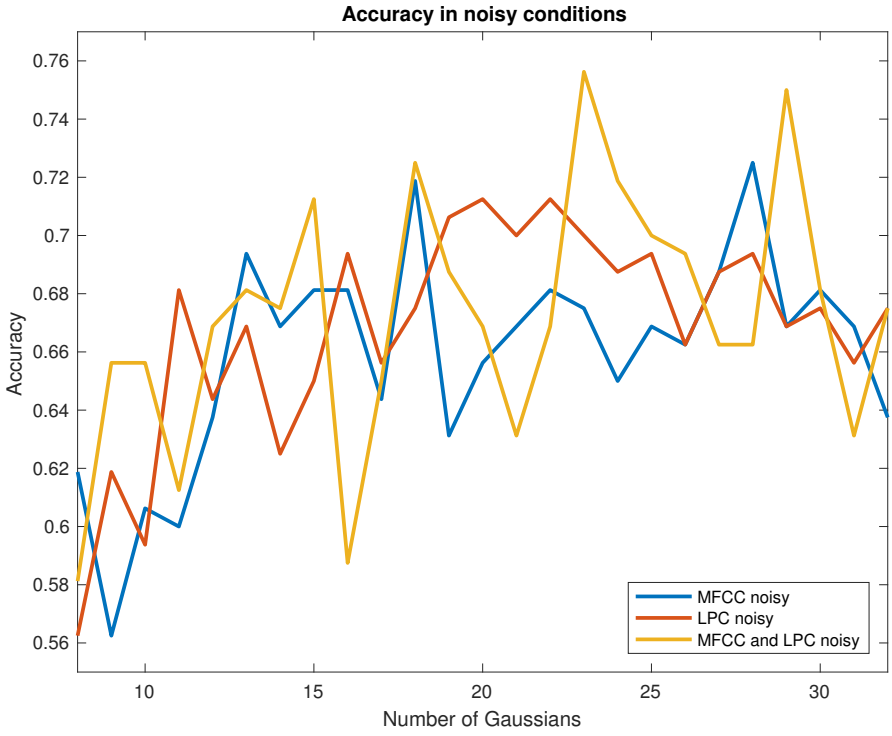

Figure 3: Accuracy in all models tested in noisy conditions for different number of Gaussian components

5 Discussion and conclusion

By investigating the confusion matrices for all models in clean conditions we observe that speaker number 13 was the most likely to be misclassified. This speaker was classified either as speaker number 1, 9 or 15. For the noisy conditions there is no clear pattern of misclassified speakers, as the accuracy are much lower in these cases.

Further the results do not give any clear indication of the preferred number of Gaussian components to use. For the clean conditions the model with both MFCC and LPC suggest the highest accuracy for 15 and 25 Gaussians. However, in the noisy conditions the maximum accuracy is given with 24 Gaussians. The results for the tests conducted in noisy conditions suggest that a higher number of Gaussians are preferred, as there is a slightly positive trend in the accuracy. However the variability is relatively large, and therefore we cannot conclude that a specific number of Gaussian components should be applied to the developed Speaker Identification models.

When comparing the three different models, none of the models show a clear tendency of outperforming the other models. All models provide similar results in terms of accuracy. In clean conditions the models give a accuracy close to 1, regardless of the number of Gaussians used. However all models suffer when testing the systems in noisy conditions. We therefore conclude that the system is highly applicable for speaker identification in clean conditions. For identification of speakers in noisy conditions the systems should be revised, to attempt to improve the accuracy of the models further.

For reducing the computation time, we restricted the number of iterations in the EM algorithm, fitting the GMM distribution, to a maximum of 500 iterations. For further work, the maximum number of iterations could be increased to investigate if this affect the performance of the identification systems. The number of model orders for LPC are set to 8. Increasing this number might also yield a higher accuracy. Finally different models for feature extraction could be investigated, in order to improve the accuracy in noisy conditions.