KTH Computer Science
and Communication

DT2118 Lab1: Feature extraction

1 Objective

The objective is to experiment with different features commonly used for speech analysis and recognition. The lab is designed in Python, but the same functions can be obtained in Matlab/Octave or using the Hidden Markov Toolkit (HTK). In Appendix A, a reference table is given indicating the correspondence between different systems.

2 Task

- compute MFCC features step-by-step
- examine features
- evaluate correlation between feature
- compare utterances with Dynamic Time Warping
- illustrate the discriminative power of the features with respect to words
- optional: perform hierarchical clustering of utterances

In order to pass the lab, you will need to follow the steps described in this document, and produce a report where you describe your work and answer the questions asked here. The report should be submitted in the Assignment section in the course Web on KTH Social `https://www.kth.se/social/course/DT2118/`. One submission should be done for each group, clearly stating all the members of that group.

3 Data

The file `tidigits_examples.npz` contains the data to be used for this exercise. The file contains two arrays: `example` and `tidigits`[1].

3.1 example

The array `example` can be used for debugging because it contains calculations of all the steps in Section 4 for one utterance. It can be loaded with:

```
import numpy as np
example = np.load('tidigits_examples.npz')['example']
```

[1]If you wish to use Matlab/Octave instead of Python, use the provided `py2mat.py` script to convert `tidigits_examples.npz` to Matlab format. Load the file with `load tidigits_examples`. You will load two cell arrays with the corresponding data stored in structures.

The element `example[0]` is a dictionary with the following keys:

`samples`:	speech samples for one utterance
`samplingrate`:	sampling rate
`frames`:	speech samples organized in overlapping frames
`preemph`:	pre-emphasized speech samples
`windowed`:	hamming windowed speech samples
`spec`:	squared absolute value of Fast Fourier Transform
`logspec`:	base 10 log of `spec`
`mspec`:	`spec` multiplied by Mel filterbank
`mfcc`:	Mel Frequency Cepstrum Coefficients

Figure 1 shows the content of the elements in `example`.

3.2 tidigits

The array `tidigits` contains a total of 44 spoken utterances from one male and one female speaker from the TIDIGITS database (`https://catalog.ldc.upenn.edu/LDC93S10`). The file was generated with the script `tidigitsCollectExamples.py`[2]. For each speaker, 22 speech files are included containing two repetitions of isolated digits (eleven words: "oh", "zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine"). You can read the file from Python with:

```
tidigits = np.load('tidigits_examples.npz')['tidigits']
```

The variable `tidigits` is an array of dictionaries. Each element contains the following keys:

`filename`:	filename of the wave file in the database
`samplingrate`:	sampling rate of the speech signal (20kHz in all examples)
`gender`:	gender of the speaker for the current utterance (`man`, `woman`)
`speaker`:	speaker ID for the current utterance (`ae`, `ac`)
`digit`:	digit contained in the current utterance (`o, z, 1, ..., 9`)
`repetition`:	whether this was the first (`a`) or second (`b`) repetition
`samples`:	array of speech samples

4 Mel Frequency Cepstrum Coefficients step-by-step

Follow the steps below to computer MFCCs. Use the `example` array to double check that your calculations are right.

4.1 Enframe

Write a Python function called `enframe`[3] that takes as input speech samples, the window length in samples and the number of samples overlap between consecutive windows and outputs a two dimensional array where each row is a window of samples. Apply the enframe function to the utterance `example[0]['samples']` with window length of 20 milliseconds and shift of 10 ms (figure out the length and shift in samples from the sampling rate). Use the `imshow` function from `matplotlib.pyplot` to plot the resulting array. This should correspond to the array in `example[0]['frames']`.

[2]The script is included only for reference in case you need to use the full database in the future. In that case, you will need to ask me for access rights to the database on AFS.

[3]for this an all following functions, you find prototypes in `proto.py`.

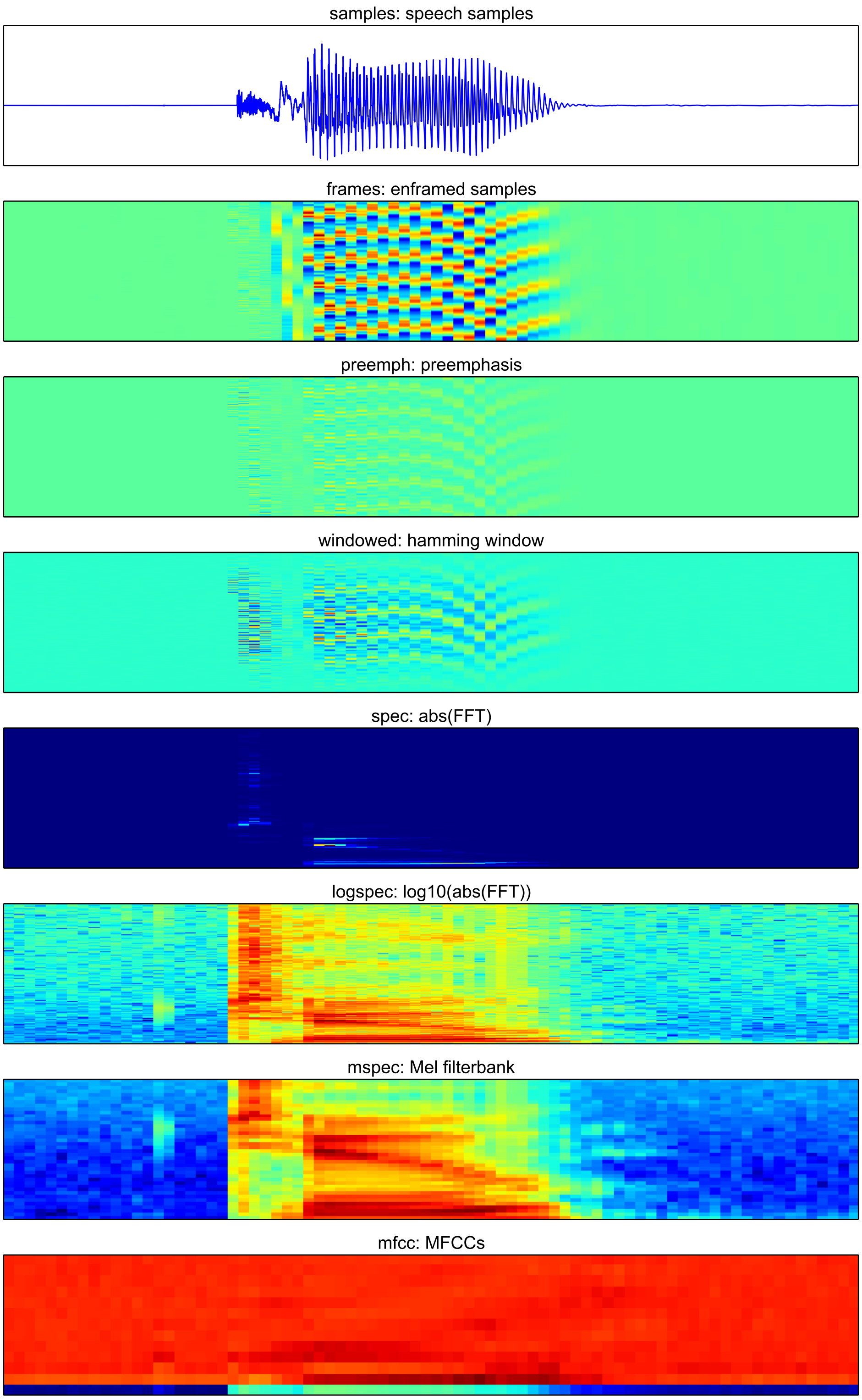

Figure 1. Evaluation of MFCCs step-by-step

4.2 Pre-emphasis

Define a pre-emphasis filter with pre-emphasis coefficient 0.97 using the `lfilter` function from `scipy.fftpack`. Explain how you defined the filter coefficients. Apply the filter to each frame in the output from the `enframe` function. This should correspond to the `example[0]['preemph']` array.

4.3 Hamming Window

Define a hamming window of the correct size using the `hamming` function from `scipy.signal`. Plot the window shape and explain why this windowing should be applied to the frames of speech signal. Apply hamming window to the pre-emphasized frames of the previous step. This should correspond to the `example[0]['windowed']` array.

4.4 Fast Fourier Transform

Compute the squared modulus of the Fast Fourier Transform (FFT) from `scipy.fftpack`, with length 512 samples, and the base 10 log of the FFT. Plot the resulting spectrogram with `imshow`. Beware of the fact that the FFT bins correspond to frequencies that go from 0 to f_{max} and back to 0. What is f_{max} in this case according to the Sampling Theorem? The two arrays should correspond to `example[0]['spec']` and `example[0]['logspec']`, respectively.

4.5 Mel filterbank

Use the `trfbank` function, provided in the `tools.py` file, to create a bank of triangular filters linearly spaced in the Mel frequency scale. Plot the filters in linear frequency scale. Describe the distribution of the filters along the frequency axis. Apply the filters to the output of the squared absolute FFT for each frame and take the base 10 log of the result. Plot the resulting filterbank outputs with `imshow`. This should correspond to the `example[0]['mspec']` array.

4.6 Cosine Transofrm

Apply the Discrete Cosine Transform (from `scipy.fftpack.realtransforms`) to the outputs of the filterbank. Use coefficients from 0 to 12 (13 coefficients). Plot the resulting coefficients with `imshow`. This should correspond to `example[0]['mfcc']`

Once you are sure all the above steps are correct, wrap the whole procedure into a `mfcc` function (`proto.py`). Compute the MFCCs for all the utterances in the `tidigits` array. Observe differences for different utterances.

5 Feature Correlation

Concatenate all the MFCC frames from all utterances in the `tidigits` array. Then compute the correlation matrix between features and display the result with `imshow`. Are features correlated? Is the assumption of diagonal covariance matrices for Gaussian modelling justified? Compare the results you obtain for the MFCC features with those obtained with the filterbank features (output of the Mel filterbank).

6 Gaussian Mixture Models

Use the concatenated MFCCs from the previous section to train a Gaussian Mixture Model with the class `GMM` from `sklearn.mixture`. Use 16 components for the mixture with diagonal covariance matrices. For each utterance in the `tidigits` array and using the trained model, compute posterior probabilities per frame and Gaussian in the model. Display the probabilities for some utterance.

7 Comparing Utterances

Write a function called `dtw` (`proto.py`) that takes as input a $[N \times M]$ distance matrix and outputs the result of the Dynamic Time Warping algorithm. The input matrix contains local distances between two sequences of length N and M respectively. The main output is the distance between two sequences (utterances), but you may want to output also the best path for debugging reasons. For each pair of utterances in the `tidigits` array:

1. compute the local Euclidean distances between MFCC vectors in the first and second utterance
2. compute the global distance between utterances with the `dtw` function you have written

Store the global pairwise distances in a matrix D. Display the matrix with `imshow`. Compare distances within the same digit and across different digits. Does the distance separate digits well even between different speakers?

Optional: run hierarchical clustering on the distance matrix D using `linkage` function from `scipy.cluster.hierarchy`. Use the "complete" linkage method. Display the results with the function `dendrogram` from the same library, and comment them. Use the `wid`s from the `tidigits` array as labels to simplify the interpretation of the results.

Repeat the same procedure using standardized data, that is remove the global mean and divide by the global standard deviation of the MFCCs. Does this improve results? Also try using GMM posteriors as features instead of MFCCs. Do you notice any difference?

A Alternative Software Implementations

Although this lab has been designed for being carried out in Python, several implementations of speech related functions are available.

A.1 Matlab/Octave Instructions

The Matlab signal processing toolbox is one of the most complete signal processing piece of software available. Many speech related functions are however implemented in third party toolboxes. The most complete are the Voicebox[4] which is more oriented towards speech technology and the Auditory Toolbox[5] that is more focused on human auditory models.

If you use Octave instead of Matlab, make sure you have the following extra packages (in parentheses are the names of the corresponding `apt-get` packages for Debian based GNU Linux distributions, all packages are already installed on CSC Ubuntu machines):

- signal (`octave-signal`)

A.2 Hidden Markov Models Toolkit (HTK)

HTK is a powerful toolkit developed by Cambridge University for performing HMM-based speech recognition experiments. The HTK package is available at all CSC Ubuntu stations, or can be download for free at `http://htk.eng.cam.ac.uk/` after registration to the site. Its manual, the HTK Book, can be downloaded separately. In spite of being open source and free of charge, HTK, is unfortunately not free software in the Free Software Foundation sense because neither its original form nor its modifications can be freely distributed. Please refer to the license agreement for more information.

The HTK commands that are relevant to this exercise are the following:

HCopy: feature extraction tool. Can read audio files or feature files in HTK format and outputs HTK format files

HList: terminal based visualization of features. Reads HTK format feature files and displays information about them

General options are:

- `-C config`: reads configuration file `conf`
- `-S filelist`: reads list of files to process from `filelist`

for a complete list of options and usage information, run the commands without arguments.

Hint: `HList -r ...`: the `-r` option in `HList` will output the feature data in raw (ascii) format. This will make it easy to import the features in other programs such as python, Matlab or R.

Table 2 lists a number of possible spectral features and the corresponding HTK codes to be used in HCopy or HList.

[4] `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html`
[5] http://amtoolbox.sourceforge.net/

Feature name	Matlab	Python
Linear filter	`filter`	`scipy.signal.lfilter`
Hamming window	`hamming`	`scipy.signal.hamming`
Fast Fourier Transform	`fft`	`scipy.fftpack.fft`
Discrete Cosine Transform	`dct`	`scipy.fftpack.realtransforms.dct`
Gaussian Mixture Model	`gmdistribution`	`sklearn.mixture.GMM`
Hierarchical clustering	`linkage`	`scipy.cluster.hierarchy.linkage`
Dendrogram	`dendrogram`	`scipy.cluster.hierarchy.dendrogram`
Plot lines	`plot`	`matplotlib.pyplot.plot`
Plot arrays	`image, imagesc`	`matplotlib.pyplot.imshow`

Table 1. Mapping between Matlab and Python functions used in this exercise

Feature name	KTH code
linear filer-bank parameters	`MELSPEC`
log filter-bank parameters	`FBANK`
Mel-frequency cepstral coefficients	`MFCC`
linear prediction coefficients	`LPC`

Table 2. Feature extraction in HTK. The HCopy executable can be used to generate features from wave file to feature file. HList can be used to output the features in text format to stdout, for easy import in other systems