521285S Affective Computing (Laboratory Exercises)

Lab – II Emotion Recognition from Speech

Objective

Your task is to extract a set of prosodic correlates (i.e. suprasegmental speech parameters) and cepstral features from speech recordings. Then, an emotion recognition system is constructed to recognize happy versus sad emotional speech (a quite easy two class problem) using a simple supervised classifier training and testing structure.

The original speech data is a set of simulated emotional speech (i.e. acted) from ten speakers speaking five different pre-segmented sentences of roughly 2-3 seconds in two different emotional states (happy and sad) totaling 100 samples.

Basic prosodic features (i.e. distribution parameters derived from the prosodic correlates) are extracted using a simple voiced/unvoiced analysis of speech, pitch tracker, and energy analysis. Another set of Mel-Frequency Cepstral Coefficients (MFCC) features are also calculated for comparison.

To produce speaker independent and content dependent emotion recognition case (i.e. while a same persons samples are not included in both training and testing sets the same sentences are spoken in both the training and the testing sets) that could correspond to a public user interface with specific commands.

Support Vector Machine (SVM) classifiers are trained. A random subset of 1/2 of the available speech data (i.e. half of the persons) is used to train the emotion recognition system, first using a set of simple prosodic parameter features and a then a classical set of MFCC derived features. The rest of the data (the other half of the persons) is then used to evaluate the performances of the trained recognition systems.

Implementation

The data and toolbox files used in this exercise can be found in the Affective computing course webpage (see the Noppa system link in the footer of this document).

Download the data file 'lab2_data.mat' containing the speech sample ('speech_sample') that is used in the feature extraction part and the pre-extracted features used in the emotion recognition part of this lab ('training_data_proso', 'testing_data_proso', 'training_data_mfcc', 'training_data_mfcc, 'training_class', and 'testing_class'). Download also the appropriate toolbox packages containing the required MATLAB functions.

- VOICEBOX toolbox 'voicebox.zip'
- MIT MEDIA LAB: Affective Computing Speech Prosody Analysis Software Tools toolbox 'SpeechProsodyAnalysisTools.zip'

Study the toolbox functions (e.g. 'getF0', 'melcepst') and the generic MATLAB functions (e.g. 'hamming', 'conv', 'resample', 'filter', 'mean', 'std', 'prctile', 'kurtosis', 'sum', 'length', 'linspace', 'trainsvm', 'svmclassify', and 'confusionmat') as they are needed in the exercise.

Required plots and outputs to pass each task and the exercise are marked with a ➢ symbol.

Download the ‘lab2_data.mat’ file containing all the needed data. Load the ‘lab2_data.mat’ file in MATLAB. Select the ‘speech_sample’ containing a raw speech waveform and do the following (Note, the sampling rate (fs) of the sample speech signal is 48 kHz):

0. Preparation

Downsample the ‘speech_sample’ from the original Fs of 48 kHz to 11.025 kHz (tip: use ‘resample’).

1. MFCC calculations using the provided sample speech signal.

Download and extract the VOICEBOX toolbox ‘voicebox.zip’ into your working directory (Remember to add into path).

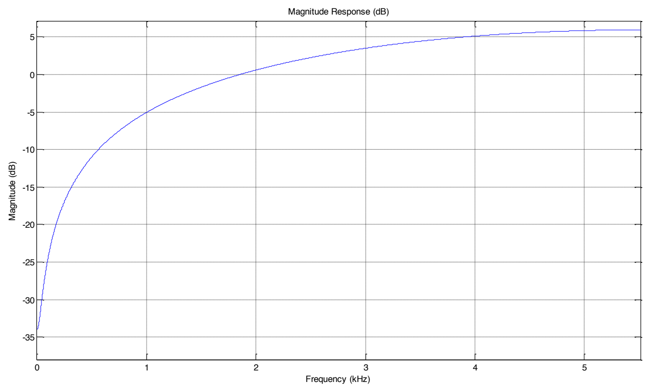
Mel-Frequency Cepstral Coefficients:

- Apply a pre-emphasis filter $H(z) = 1 - \alpha z^{-1}$ with $\alpha = 0.98$ to emphasize higher frequencies in your downsampled speech signal (Tip: use ‘filter’).
  Hint for defining the filter: you will provide two vectors **b** and **a** to define the filter, **a** for the denominator and **b** for the numerator. So finally your filter will be defined as

$$H(z) = \frac{\boldsymbol{b}[0]z^0+\boldsymbol{b}[1]z^{-1}+\cdots\boldsymbol{b}[i]z^{-i}+\cdots}{\boldsymbol{a}[0]z^0+\boldsymbol{a}[1]z^{-1}+\cdots\boldsymbol{a}[i]z^{-i}+\cdots},$$

  please note that in Matlab indices start from 1.

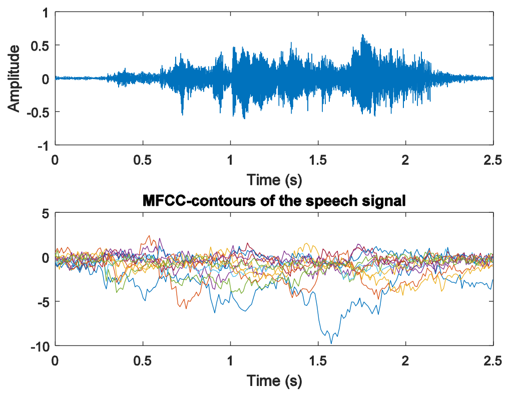
➢ Plot the pre-emphasis filter frequency response in Hz using the filter visualization tool ‘fvtool’ (use the correct sampling frequency!). Should be like this:



- Use ‘melcepst’ (use the default parameters) to create 12 MFCC feature contours.

➢ Plot the speech sample waveform (subplot 1) and all the raw MFCC contours (in the same subplot 2) with respect to time in a 2x1 subplot matrix. Remember also to put titles in the plots (including the axis). Note: the default frame increment length is 256/2=128 samples, corresponding roughly to 11.6ms frames at Fs=11025Hz. Tip: You can use ‘linspace’ to easily make the corresponding time vectors.

The material used in this exercise can be found in the course web page at https://noppa.oulu.fi/noppa/kurssi/521285s/harjoitukset.



Extract the following utterance level MFCC features (12 in total):

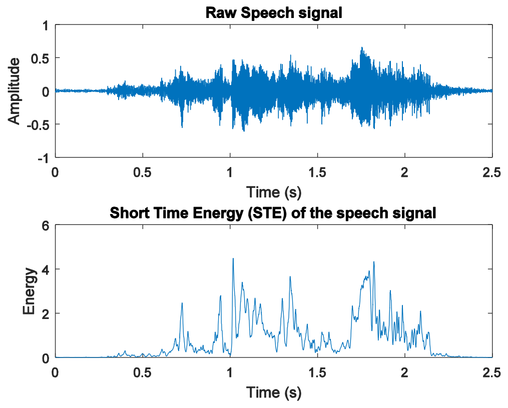
- Calculate the 'mean' of each MFCC feature contour to produce a total of 12 features.
- ➢ Check that the means are same as in the first row of 'training_data_mfcc' matrix.

2. Perform prosodic parameter extraction using the provided sample speech signal.

Download and extract the MIT MEDIA LAB: Affective Computing Speech Prosody Analysis Software Tools toolbox 'SpeechProsodyAnalysisTools.zip' into your working folder (Remember to add into path all the subfolders).

Intensity/Energy:

- Calculate the short time energy (STE) of the downsampled 'speech_sample' using the squared signal $x(t)^2$ and a 0.01s hamming window frames. Tip: use 'conv' to apply the window (Note! the extra length of the window. Clip half a window length from the beginning and at the end).
- ➢ Plot the speech sample waveform and the STE contour with respect to time in a 2x1 subplot matrix. Remember also to put titles in all plots (including the axis).



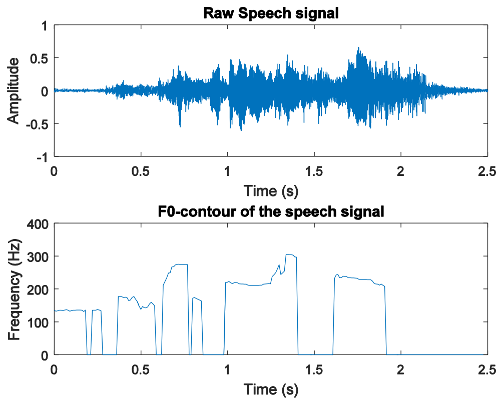
Extract the following prosodic features:

- Calculate 5 distribution parameter features from the utterance; the mean 'mean', Standard Deviation (SD) 'std', 10% percentile 'prctile', 90% percentile 'prctile', and kurtosis 'kurtosis' of the STE.

The material used in this exercise can be found in the course web page at https://noppa.oulu.fi/noppa/kurssi/521285s/harjoitukset.

Pitch/F0:

- Calculate the F0 contour of your downsampled speech signal using the ‘getF0’ function in 0.01s frames.
- ➢ Plot the speech sample waveform and the pitch contour with respect to time in a 2x1 subplot matrix. Remember also to put titles in all plots (including the axis).



Extract the following prosodic features:

- Calculate 5 distribution parameter features from the **<u>Voiced Part of the</u>** utterance; the mean ‘mean’, Standard Deviation (SD) ‘std’, 10% percentile ‘prctile’, 90% percentile ‘prctile’, and kurtosis ‘kurtosis’ of the **Voiced** F0 contour (i.e. F0 > 0).

Rhythm/Durations:

- Perform a Voiced/Unvoiced segmentation of the speech signal by using the ‘getF0’ method. Tip: find the voiced frames (i.e. F0 > 0)
- Calculate each Voiced and Unvoiced segment lengths (Note! Unvoiced frames are marked with 0 F0 values). (Tip: use ‘diff’ to get segment boundaries and ‘find’ to get the frame positions where segments change)

Extract the following prosodic features:

- From the segmentation, calculate the means and SDs of both Voiced and Unvoiced segment lengths (i.e. voiced segment mean length, SD of voiced segment lengths, unvoiced segment mean length, SD of unvoiced segment lengths).
- Calculate also the voicing ratio, i.e. the ratio of voiced segments versus total segments (Tip: You can do this simply by using the frames).

Check the prosodic features:

- ➢ Store the features (15 in total) in a vector in the following order: F0 mean, SD of F0, 10% percentile of F0, 90% percentile of F0, kurtosis of F0, STE mean, SD of STE, 10% percentile of STE, 90% percentile of STE, kurtosis of STE, voiced segment mean length, SD of voiced segment lengths, unvoiced segment mean length, SD of unvoiced segment lengths, and voicing ratio. Check the feature vectors by comparing your values with the first row of ‘training_data_proso’ matrix.

The material used in this exercise can be found in the course web page at https://noppa.oulu.fi/noppa/kurssi/521285s/harjoitukset.

3. **Train SVM classifiers**.

Use the ‘svmtrain’ function to train Support Vector Machine (SVM) classifiers. The ‘training_data_proso’ and ‘training_data_mfcc’ matrixes contain the calculated prosodic features for the training set (9 features in each row representing a speech sample) and MFCC derived features (12 features) respectively. The ‘training_class’ group vector contains the class of samples: 1 = happy, 2 = sad; corresponding to the rows of the training data matrices.

- Train an SVM with the prosody data using the ‘training_data_proso’ features and a 3rd order polynomial kernel.
- Train an SVM with the MFCC data using the ‘training_data_mfcc’ features and a 3rd order polynomial kernel.

4. **Test the classifiers and plot confusion matrices.**

Use the ‘svmclassify’ function (and your trained SVM structures) to classify the ‘training_data_*’ and the ‘testing_data_*’ data matrices. Then, calculate average classification performances for both training and testing data. The correct class labels corresponding with the rows of the training and testing data matrices are in the variables ‘training_class’ and ‘testing_class’, respectively.

- ➢ Calculate the average classification performances for the training data (‘training_data_proso’ and ‘training_data_mfcc’) using the corresponding prosody and MFCC trained SVMs.
- ➢ Calculate the average classification performance for the testing data (‘testing_data_proso’ and testing_data_mfcc’) using the corresponding prosody and MFCC trained SVMs.

Plot confusion matrices for the training and testing data for both classifiers. Tip, use ‘confusionmat’ function.

- ➢ Calculate the confusion matrix of the prosody trained SVM using the ‘training_data_proso’.
- ➢ Calculate the confusion matrix of the prosody trained SVM using the ‘testing_data_proso’.
- ➢ Calculate the confusion matrix of the MFCC trained SVM using the ‘training_data_mfcc’.
- ➢ Calculate the confusion matrix of the MFCC trained SVM using the ‘testing_data_mfcc’.

The material used in this exercise can be found in the course web page at https://noppa.oulu.fi/noppa/kurssi/521285s/harjoitukset.

5. Optional task: Cross-validation estimation of recognition performance (NOT REQUIRED TO PASS THE EXERCISE).

Generate a person independent 10-fold cross-validation (CV) estimate of the emotion recognition system performance.

- Join the training/testing data matrices and the class vectors. Combine also the ‘training_data_personID’ and ‘testing_data_personID’ vectors that are needed to make the CV folds.
- Construct the CV folds by training ten SVMs. For each SVM nine persons’ data is used as the training set (i.e. 90 samples) and one persons’ samples are kept as the test set (i.e. 10 samples) for the respective fold (i.e. each SVM has different persons’ samples excluded from the training set). Test each ten trained SVMs by using the corresponding one held-out persons’ samples and then calculate the average classification performances for each fold.
- Calculate the mean and SD of the ten CV fold performances to produce the final CV performance estimate of the emotion recognition system.

The material used in this exercise can be found in the course web page at https://noppa.oulu.fi/noppa/kurssi/521285s/harjoitukset.