Description of the New Matlab Code of Hierarchical Decision Tree for Large Population Speaker Identification

* The code implements the fuzzy-clustering-based hierarchical decision tree for large population speaker identification. I put all the codes in the folder “code” which consists of two sub-folders and some M-files. The two sub-folders are “PitchTrack” and “GMM”. They contain the source code written in C. All the other files are M-files used in Matlab.
* All codes can be divided into three categories: code of feature extraction for clustering, code of decision tree construction and GMM code used at leaf node.

**Brief System Description**

* We derived a total of six features for clustering and construct a six-level decision tree, accordingly. From the first level to the sixth level, pitch feature, PSNR feature of positive pulses in LP residual signal, Skewness feature of positive pulses in LP residual signal, PSNR feature of negative pulses in LP residual signal, Skewness feature of negative pulses in LP residual signal and Width feature of positive pulses in LP residual signal are adopted successively. The clustering is carried out from upper levels to lower levels until the leaf nodes are reached. At a certain level, if a node already contains a sufficient small number of speakers (e.g.<20), we treat it as a leaf node and there will not be any further clustering conducted for this node at lower levels. At the leaf node, MFCC+GMM+ML are applied for final speaker recognition. More details of system description, decision tree construction, fuzzy clustering algorithm design and feature extraction for clustering can be found in the paper.

**Code Description**

* **Part-I Code of Feature Extraction**
* For pitch feature extraction, you can use the C code in folder “PitchTrack”. After compiling all the codes in the folder, you should be able to obtain the executive file “PitchTrack\_all”. The usage of PitchTrack\_all is: PitchTrack\_all *arg1 arg2*, where arg1 is the full name of the directory containing all the training data of all speakers and arg2 denotes the directory storing all pitch estimation of all speakers. Here is an example: PitchTrack Data\_Train PitchFea\_Train. The folder “Data\_Train”should contain sub folders each of which named after the speaker’s name and each sub folder contains all the speech samples of the corresponding speaker. After running the code, under the folder “PitchFea\_Train”, every speaker has a folder named after the speaker’s name and two .txt files are stored in the folder. Raw\_Pitch.txt contains all the pitch frequency values without any processing. Pitch.txt contains the processed pitch frequency values that are used for clustering. We will use Raw\_Pitch.txt for the speaker identification task.
* For positive-pulse PSNR feature extraction, you can use the PSNR\_Pos.m file. The usage of the M-file is: PSNR\_Pos( *arg1, arg2*), where arg1 is the full name of the directory containing all the speech samples of all speakers and arg2 denotes the directory storing all positive pulse PSNR estimations of all speakers. Here is an example: PSNR\_Pos(‘../data/Data\_Train’,’Fea\_PSNR\_Pos’). The folder ‘../data/Data\_Train’should contain sub folders each of which named after the speaker’s name and each sub folder contains all the speech samples of the corresponding speaker. After running the code, under the folder ‘Fea\_PSNR\_Pos’, every speaker has a folder named after the speaker’s name and a re\_fea.txt file is stored in the folder. The re\_fea.txt file contains all the positive pulse PSNR feature estimations.
* For negative-pulse PSNR feature extraction, you can use the PSNR\_Neg.m file. The usage of the M-file is the same as PSNR\_Pos.m file.
* For positive-pulse Skewness feature extraction, you can use the Skewness\_Pos.m file. The usage of the M-file is the same as PSNR\_Pos.m file.
* For negative-pulse Skewness feature extraction, you can use the Skewness\_Neg.m file. The usage of the M-file is the same as PSNR\_Pos.m file.
* For positive-pulse Width feature extraction, you can use the Width\_Pos.m file. The usage of the M-file is the same as PSNR\_Pos.m file.
* The other two M-files “findpeaks” and “utter” are called by the M-files described above. "findpeaks” detects peaks in a signal and “utter” determined the voiced speech frames for a given input speech.
* **Part-II Code of Decision Tree Construction**
* For the construction of the decision tree, you can use the 6L\_Tree.m file. The function takes 13 parameters as the inputs. They are listed as follows:

1. The full name of the directory containing all the training data of all speakers

2.The full name of the directory storing all pitch estimations of all speakers from training samples

3. The full name of the directory storing all positive pulse PSNR feature estimations of all speakers from training samples

4. The full name of the directory storing all negative pulse PSNR feature estimations of all speakers from training samples

5. The full name of the directory storing all positive pulse Skewness feature estimations of all speakers from training samples

6. The full name of the directory storing all negative pulse Skewness feature estimations of all speakers from training samples

7. The full name of the directory storing all positive pulse Width feature estimations of all speakers from training samples

8. The full name of the directory storing all pitch estimations of all speakers from testing samples

9. The full name of the directory storing all positive pulse PSNR feature estimations of all speakers from testing samples

10. The full name of the directory storing all negative pulse PSNR feature estimations of all speakers from testing samples

11. The full name of the directory storing all positive pulse Skewness feature estimations of all speakers from testing samples

12. The full name of the directory storing all negative pulse Skewness feature estimations of all speakers from testing samples

13. The full name of the directory storing all positive pulse Width feature estimations of all speakers from testing samples

After you running the code, you should be able to see two new directories: “Train\_6L” and “Test\_6L”. The “Train\_6L” contains sub directories representing all leaf nodes of the tree. Each sub directory has been assigned a unique name. Each sub directory contains all speakers who are clustered into the leaf node it represents in the training phase. There may be common speakers between different sub directories because of the fuzzy clustering we adopt. The “Test\_6L” contains sub directories representing all leaf nodes that at least one speaker under test is determined to belong to in the testing phase. These two directories will be used by our MFC+GMM module for speaker identification. Two important values regarding the performance of clustering (i.e. the average population rate of the leaf node to the root node and the clustering accuracy at the leaf-node level of the tree) will also be shown in the screen after running the M-file.

* **Part-III Code of MFCC+GMM at Leaf Nodes for Speaker Identification**
* For the identification by using MFCC+GMM at leaf nodes, you can use the code in the directory “GMM”. “GMM” contains an executive file “wave2feat” and two sub directories “GMM\_Train” and “GMM\_Test”.

**GMMTrain**

* **Function: MFCC feature extraction + GMM training for all enrolled speakers**
* **Usage:**

GMMTrain *arg1 arg2 arg3* (source code is in “GMM\_Train”)

* **Parameter Description:**

*arg1* is the full name of the directory containing all the training data of all speakers (e.g. there are N speakers). It should contain N sub folders each of which named after the speaker’s name.

*arg2* denotes the directory containing MFCC features and GMM parameters of all speakers. Under the directory, each speaker will have a folder named after the speaker. In the folder, the intermediate files containing MFCC feature, GMM parameters, computational complexity, etc. are stored.

*arg3* is the parameter for frame selection used in MFCC feature extraction. For MFCC extraction, we need to determine which frame is speech frame and which frame is silence frame. Only the speech frame will be used for MFCC extraction. *arg3* denotes the threshold that is used for speech frame determination. If the energy of a frame is larger than arg8, then this frame is considered as speech frame. Otherwise, it is believed to be silence frame and thus will be dropped.

* **Usage example:**

GMMTrain C:\Demorun\Data\_Train Proc\_Train 0.0001

**GMMTest**

* **Function: Use GMM + Maximum Likelihood to determine the speaker’s identity for the input speech at only the enabled leaf node.**
* **Usage:**

GMMTest *arg1 arg2 arg3 arg4 arg5* (source code is in “GMM\_Test”)

* **Parameter Description:**

*arg1* is the full name of the directory containing all the testing data of all speakers (e.g. there are N speakers). It should contain N sub folders each of which named after the speaker’s name.

*arg2* is“Test\_6L” generated by the M-file “6L\_Tree” which is responsible for constructing the hierarchical decision tree.

*arg3* denotes denotes the directory containing MFCC features of all testing speakers. Under the directory, each speaker will have a folder named after the speaker. In the folder, MFCC feature are stored.

*arg4* is“Train\_6L” generated by the M-file “6L\_Tree” which is responsible for constructing the hierarchical decision tree.

*arg5* is the parameter for frame selection used in MFCC feature extraction. Please refer to *arg3* of GMMTrain.

* **Usage example:**

GMMTest C:\Demorun\Data\_Test ./ Test\_6L ./ Proc\_Test ./Train\_6L 0.0001

**wave2feat**

* **Function: Do MFCC Feature Extraction**
* **Description:**

wave2feat works to calculate mel-frequency cepstral coefficients (MFCC) from recorded wave files. With default settings, wave2feat gives errors if the input wave file has sample frequency greater than 20 kHz. It is also suggested to convert recorded speeches into 1-channel wave files. When running GMMTest, make sure the two executive files “wave2feat” and “GMMTest” are in the same directory.

**Some Comments**

* I strongly suggest you do not include spaces or other strange characters in any of the folder/file names.
* After running the GMMTest, you should be able to see a new txt file recording the result of the identification. At each line of the file, the name of the speaker under test is at the left and the name of the speaker our system gives is at the right. If there is a match, it’s a correct identification; Otherwise, it’s an error.
* The current code is designed for performance evaluation rather than practical applications of speaker identification, i.e. given an input, output the speaker identity determined by the system. We need to revise the code a bit to make it work like an identification system in the real applications.