Real-Time Text-Independent Speaker Identification System

IMPLEMENTED BY USING PYAUDIO

He Huang, Shihong Fang | DSP LAB | December 17, 2015

Table of Contents

1. Basic Idea:

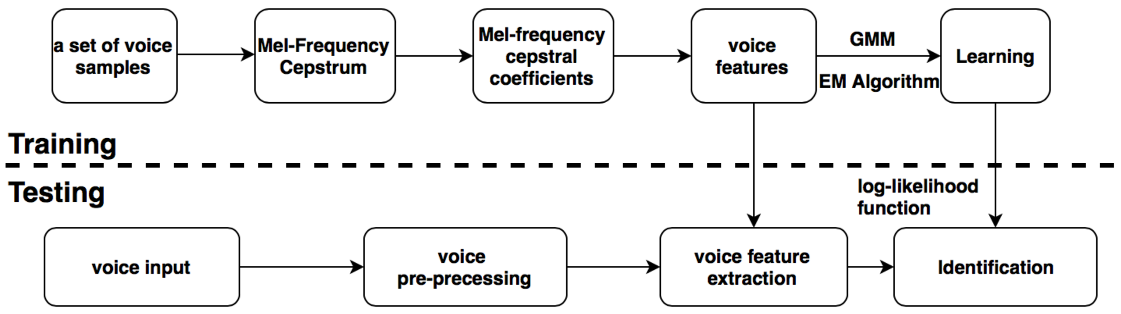

Figure 1. The Diagram of the Entire System

In speaker identification, most of the computation originates from the distance or likelihood computations between the feature vectors of the unknown speaker and the models in the database. The identification time depends on the number of feature vectors, their dimensionality, the complexity of the speaker models and the number of speakers.

In this project, we focus on achieving Real-Time Speaker Identification based on GMM modeling method and compute the log-likelihood function to decide the person who is now speaking.

And we partition our project into two parts. The first is training part and the second is test part. When executing the python file on command line, the computer will ask speaker's name and training GMM model for them after all speakers' voice are imported. And after training process completed, the computer can test the speaker who's now speaking.

2. Important Concepts:

2.1 Mel-Frequency Cepstral Coefficients (MFCCs)

- The **Mel-Frequency Cepstrum** (**MFC**) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency.

- **Mel-frequency cepstral coefficients** (**MFCCs**) are coefficients that collectively make up an MFC. The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the Mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum.

- **Why do we use MFCCs?**
  The main point to understand about speech is that the sounds generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. This shape determines what sound comes out. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope.

2.2 Gaussian Mixture Model (GMM)

- A **Gaussian Mixture Model (GMM)** is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMMs are commonly used as a parametric model of the probability distribution of continuous measurements or features in a biometric system, such as vocal-tract related spectral features in a speaker recognition system. GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm or Maximum A Posteriori (MAP) estimation from a well-trained prior model.

- **Model Description**

$$p(\vec{x}|\lambda) = \sum_{i=1}^{M} p_i b_i(\vec{x})$$

where $\vec{x}$ is a D-dimensional random vector, $b_i(\vec{x}),\ i = 1, \dots, M,$ are the component densities and $p_i,\ i = 1, \dots, M,$ are the mixture weights. Each component density is a D-variate Gaussian function of the form

$$b_i(\vec{x}) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} exp\left\{-\frac{1}{2}(\vec{x} - \vec{\mu_i})'\Sigma_i^{-1}(\vec{x} - \vec{\mu_i})\right\}$$

with mean vector $\vec{\mu_i}$ and covariance matrix Σ_i. The mixture weights satisfy the constraint that

$$\sum_{i=1}^{M} p_i = 1$$

The complete Gaussian mixture density is parameterized by the mean vectors, covariance matrices and mixture weights form all component densities. These parameters are collectively represented by the notation

$$\lambda = \{p_i, \mu_i, \Sigma_i\}\ i = 1, \dots, M.$$

For speaker identification, each speaker is represented by a GMM and is referred to by his/her model λ.

The GMM can have several different forms depending on the choice of covariance matrix. The model can have one covariance matrix per Gaussian component (called nodal covariance), one covariance matrix for all Gaussian components in a speaker model (grand covariance), or a single covariance matrix shared by all speaker models (global covariance). In our project, we use the diagonal nodal covariance matrix.

- **Why we use GMM to represent speaker model**

First, the individual component densities of a multi-modal density, like the GMM, may model some underlying set of acoustic classes. It is reasonable to assume the acoustic space corresponding to a speaker's voice can be characterized by a set of acoustic classes representing some broad phonetic events, such as vowels, nasals, or fricatives. These acoustic classes reflect some general speaker-dependent vocal tract configurations that are useful for characterizing speaker identity. The spectral shape of the ith acoustic class can in tum be represented by the mean $\vec{\mu_i}$ of the ith component density, and variations of the average spectral shape can be represented by the

covariance matrix Σ_i;. Because all training or testing speech is unlabeled, the acoustic classes are "hidden" in that the class of an observation is unknown. Assuming independent feature vectors, the observation density of feature vectors drawn from these hidden acoustic classes is a Gaussian mixture.
Secondly, Gaussian mixture densities for speaker identification is the empirical observation that a linear combination of Gaussian basis functions is capable of representing a large class of sample distributions. One of the powerful attributes of the GMM is its ability to form smooth approximations to arbitrarily-shaped densities.

3. Learning Model:

In learning model, we extract human voice features and build models for each person based on the features extracted. The features we used here are called Mel-Frequency Cepstral Coefficients (Abbreviated as MFCCs) and the model we build for reach person is called Gaussian Mixture Model. (Abbreviated as GMM).

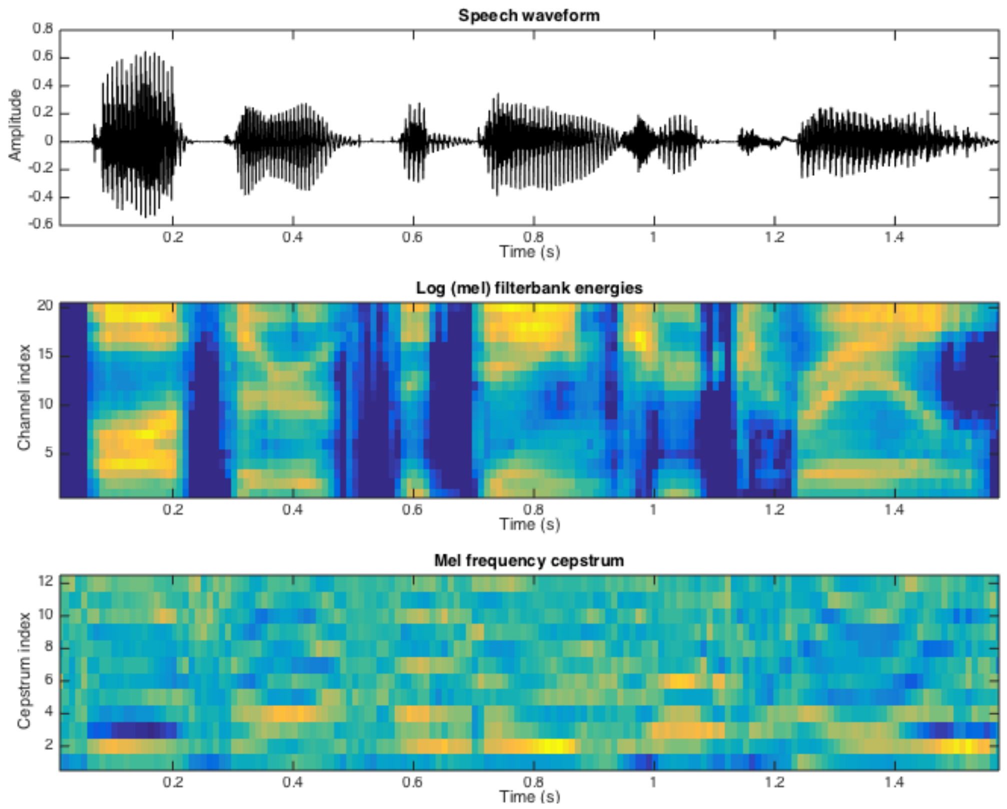

Figure 2. The waveform of sample voice(author.wav) and its corresponding MFCCs

3.1 The Procedure to Extract MFCC

1) Pre-Emphasis

In this step isolated word sample is passed through a filter which emphasizes higher frequencies. It will increase the energy of signal at higher frequency.

2) **Frame Blocking**
The speech signal is segmented into small duration blocks of 20-30 ms known as frames. In our project, the duration blocks is 20ms. Voice signal is divided into N samples and adjacent frames are being separated by M (M<N). Typical values for M=100 and N=256. Framing is required as speech is a time varying signal but when it is examined over a sufficiently short period of time, its properties are fairly stationary. Therefore short time spectral analysis is done.

3) **Hamming Windowing**
Each of the above frames is multiplied with a hamming window in order to keep continuity of the signal. So to reduce this discontinuity we apply window function. Basically the spectral distortion is minimized by using window to taper the voice sample to zero at both beginning and end of each frame.

4) **Fast Fourier Transform and get Power Spectral**
FFT is a process of converting time domain into frequency domain. To obtain the magnitude frequency response of each frame we perform FFT. By applying FFT the output is power spectrum using periodogram estimate method. To implement this, we do the following:

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-\frac{j2\pi kn}{N}} \qquad 1 \le k \le K$$

Where $h(n)$ an N sample long analysis hamming window and K is the length of the FFT. The periodogram-based power spectral estimate for the speech frame

$$s_i(n)$$

is given by:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2$$

And in our program, we take 512 point FFT and keep only the first 257 coefficients.

5) **Apply the Mel filterbank to the power spectra, sum the energy in each filter**
We apply a set of 40 triangular filters to the periodogram power spectral estimate from last step. Then we multiply each filterbank with the power spectrum, then add up the coefficients. So after that, we are left with 40 numbers that give us an indication of how much energy was in each fiterbank.

6) **Take logarithm of all filterbank energies**

7) **Take the DCT of the log filterbank energies**
Take the Discrete Cosine Transform (DCT) of the 40 log filterbank energies to give 40 cepstral coefficients. For our project, we select the lower 13 of the 40 coefficients kept.

8) **Calculate the MFCC trajectories and appending to original feature vector (we do not add to our MFCC extraction procedure)**
Because the MFCC feature vector only describes the power spectral envelope of a single frame, we need to add dynamics information of the voice to the final feature vector to increase performance. The following is how they are implemented:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

Where d_t is a delta coefficient, from frame t computed in terms of the static coefficients to c_{t+n}

c_{t-n}

Typically, N is 2.

3.2 The Procedure to Get GMM Model Parameters

The main idea to get GMM model parameters of each person is to use Expectation-Maximization (EM) algorithm to estimate.

For example, we get a sequence of T training vectors $X = \{\overrightarrow{x_1}, \dots, \overrightarrow{x_T}\}$, the GMM likelihood can be written as

$$p(X|\lambda) = \prod_{t=1}^{T} p(\overrightarrow{x_t}|\lambda)$$

And the basic idea of the EM algorithm is, beginning with an initial model λ, to estimate a new model $\bar{\lambda}$, such that $p(X|\bar{\lambda}) \geq p(X|\lambda)$. The new model then becomes the initial model for the next iteration and process is repeated until some convergence threshold is reached. The following estimation formula are used which can guarantee a monotonic increase in the model's likelihood value:

Mixture Weights:

$$\overline{p_i} = \frac{1}{T}\sum_{t=1}^{T} p(i|\overrightarrow{x_t}, \lambda)$$

Means:

$$\overline{\overrightarrow{\mu_i}} = \frac{\sum_{t=1}^{T} p(i|\overrightarrow{x_t}, \lambda)\overrightarrow{x_t}}{\sum_{t=1}^{T} p(i|\overrightarrow{x_t}, \lambda)}$$

Variances:

$$\overline{\sigma_i}^2 = \frac{\sum_{t=1}^{T} p(i|\overrightarrow{x_t}, \lambda)\overrightarrow{x_t}^2}{\sum_{t=1}^{T} p(i|\overrightarrow{x_t}, \lambda)} - \overline{\mu_i}^2$$

where $\overline{\sigma_i}^2$, x_t and μ_i refer to arbitrary elements of the vectors $\overline{\sigma_i}^2$, $\overrightarrow{x_t}$ and $\overrightarrow{\mu_i}$ respectively.

The *a posteriori* probability for acoustic class i is given by

$$p(i|\overrightarrow{x_t}, \lambda) = \frac{p_i b_i(\overrightarrow{x_t})}{\sum_{k=1}^{M} p_k b_k(\overrightarrow{x_t})}$$

4. Testing Model:

In test model, we feed the computer speaker voice and then computer compute the log-likelihood function to determine the speaker who's speaking.

First, we build a database containing a group of S speakers $S = \{1, 2, \dots, S\}$ is represented by GMM's $\lambda_1, \lambda_2, \dots, \lambda_S$. And our objective is to determine $\hat{S}$ which can be interpreted as

$$\hat{S} = \arg \max_{1 \le k \le S} \Pr(\lambda_k | X) = arg \max_{1 \le k \le S} \frac{p(X|\lambda_k)\Pr(\lambda_k)}{p(X)}$$

And we assume equally likely speakers and p(X) is the same for all speaker models. The classification can be simplified as

$$\hat{S} = \arg \max_{1 \le k \le S} \Pr(X|\lambda_k)$$

Using logarithms, we can write as

$$\hat{S} = \arg \max_{1 \le k \le S} \sum_{t=1}^{T} \log p(\overrightarrow{x_t}|\, \lambda_k)$$

In which $p(\overrightarrow{x_t}|\lambda_k)$ is given in previous formula.

5. Issues and Improvement:

5.1 Initialization Issues and Methods to solve [7]

In fact, before we implement EM algorithm, we should initialize the starting model λ_0 because EM algorithm is guaranteed to find the local maxima likelihood model and GMM has several local maxima. And according to many researches, the elaborate initialization schemes are not necessary for training Gaussian mixture speaker models.

But in order to do not lose generality, our project use k-cluster method to initialize means, nodal, diagonal covariance matrix and mixture weights.

5.2 Variance Limiting and Methods to solve [7]

According to many papers we read, too small variances produce a singularity in the model's likelihood function and can degrade identification performance. The noisy data can contain outliers in the data that give rise to components with very small variances.

To avoid these spurious singularities, a variance limiting constraint is applied.

$$\overline{\sigma_i}^2 = \begin{cases} \sigma_i^{\,2}, & \sigma_i^{\,2} > \sigma_{min}^{\,2} \\ \sigma_{min}^{\,2}, & \sigma_i^{\,2} \le \sigma_{min}^{\,2} \end{cases}$$

Generally, we set$\sigma_{min}^{\,2} = 0.01$ *or* $\sigma_{min}^{\,2} = 0.1$, and in our project, we set as 0.01.

5.3 Model Order Selection [7]

Determining the number of component M in a mixture needed to model a speaker adequately is an important but difficult problem. According to the result by Prof Douglas A. Reynolds 's result, our project test time is 30s, and the optimized model selection for M is 15 or 16.

5.4 Methods for Spectral Variability Compensation [7]

From the results by Prof Douglas A. Reynolds, there are three methods to compensate spectral variability, they are Frequency Warping, Spectral Shape Compensation and use of cepstrum difference coefficients or called Delta Energy. According to the Identification Performance for different spectral variability compensation techniques applied to telephone speech which is illustrated by one of his paper. The most effective method for minimizing the channel variation effects is mean normalization which can improve the performance by an average 28% compared with non-improvement identification performance.

And here, in our project, we assume the computer's channel is more likely a static time-variant channel. So we can use Mean-Normalization to increase the identification performance.

Mean-Normalization Implementation:

$$\vec{z} = \vec{x} + \vec{h}$$

Here, $\vec{z}$ is the observed cepstral vector, $\vec{h}$ is the channel filter cepstral filter and $\vec{x}$ is the input cepstral vector. The global average vector is

$$\vec{m} = \frac{1}{T}\sum_{t=1}^{T} \vec{z_t}$$

And the final channel compensated vectors are given by

$$\vec{z_t}^{comp} = \vec{z_t} - \vec{m}$$

5.5 Methods to Improve the Test Performance ----- Feature Reusing or Row Extension (From us)

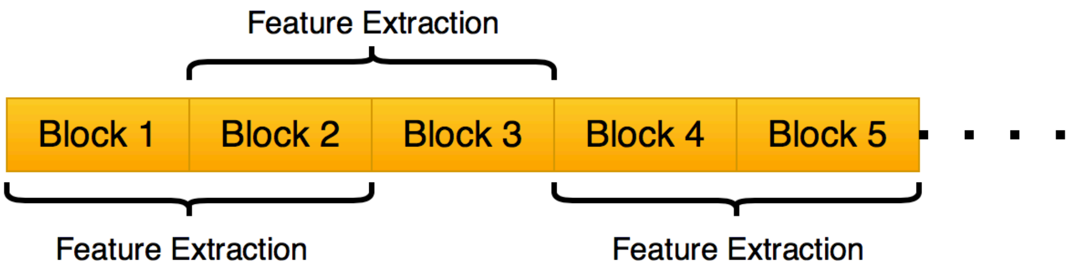

Figure 3. The Diagram of our improvement method

In Real-Time Implementation, the identification time gap is about 0.4535s, considering it's too short and the features vectors' number in this short period of time is less than 50, we propose an idea to improve the identification performance.

The idea is that we build a buffer that stores the previous block feature vectors and append them with the current block feature vectors. And the final feature we feed to the computer is doubled than that without this procedure. We call this method Feature Reusing or Row Extension.

$$X_{new} = [X_{previous}, X_{current}] along\ row\ of\ the\ feature\ matrix$$

In this way, we can double the number of feature vectors so that our identification rate can be boosted in a short period of time.

6. Performance Evaluation

6.1 Test Environment and Preparation

The programming language we use is Python of version 2.7.10 and the package we use is **sklearn.mixture.GMM** and **scipy.stats.multivariate_normal** as well as **numpy, pyaudio**. The program is tested on 15-inch MacBook Pro made in Mid 2015. In our project, we set the training time to 30 seconds and test time to 60 seconds, and our database could hold 10 persons' training voice file. However, in real test we limit the testing pool to 2 people(us). And our training material is from a randomly chosen essay *What I Have Lived For* by Bertrand Russell. Our test materials are same each time and are almost 60 seconds in length for the purpose of comparison. And in order to avoid the spectral shaping imposed by different channels, we use mono channel to collect training wav file.

6.2 Performance Evaluation

The command line will prompt the speaker's name based on the system we design. In our real test, the command line will prompt the names for 43 times within 60 seconds of testing time. Because we have already known the sequence of person, if someone's identification process is wrong at a certain period of time, after test time is out, we could compute every person's identification correction rate CR_i and it can be illustrated as flowing formula:

$$CR_i = \frac{The\ number\ of\ correct\ name}{43} \times 100\%$$

Then, we compute the average correction rate CR to get the final correction rate.

And in test procedure, we conducted 5 times test experiment. Finally, we compute final estimate correction rate C of our speaker identification system.

$$\mathrm{CR} = \frac{\overline{CR_1} + \overline{CR_2} + \cdots + \overline{CR_5}}{5}$$

Our final result is CR = 94.88%.

In fact, most errors occur when someone change to another one and because our identification time gap is too short, we think errors could not completely be avoided when transition occurred. And then we ignore the error occurred during transition about 1 second. After that, we find the final CR' = 97%.

7. Analysis

Just like we said, the errors mostly occur when transition from one person to another different person. The reason we infer is that we do not use cepstrum difference coefficients or called Delta Energy which can add the dynamic information that generally lose in normal cepstral coefficients. Because the channel actually is time-variant, but in short test time period of same people, we think the channel could be seen as time-invariant and dynamic information is not very important.

While in transition, the channel itself may not change too much but the noise and the voice source change a lot. So we can say when transition occurs, dynamic information must be considered and captured to improve the identification performance.

8. Conclusion

The reason we choose this topic is that both of us are very interested in Machine Learning Algorithm and we want to use some of them to solve practical issues. And the machine learning algorithm we use in this speaker identification system are GMM model and EM algorithm which is very efficient to estimate mixture model parameters.

In addition, we are pleased that we have accomplished this project and the result is far beyond our expectation, although there are errors during test especially transition between two speakers. But considering our identification test time gap is just 0.4535s, we think the test accuracy is relatively high.

9. Reference

[1] Bishop, Christopher M. Pattern Recognition And Machine Learning. New York: Springer, 2006. Print.

[2] Huang, Xuedong, Alejandro Acero, and Hsiao-Wuen Hon. Spoken Language Processing. Upper Saddle River, NJ: Prentice Hall PTR, 2001. Print.

[3] Kinnunen, T., E. Karpov, and P. Franti. "Real-Time Speaker Identification And Verification". IEEE Transactions on Audio, Speech and Language Processing 14.1 (2006): 277-288. Web.

[4] Lyons, James. "Mel Frequency Cepstral Coefficient (MFCC) Tutorial". Practicalcryptography.com. N.p., 2013. Web. 24 Nov. 2015.

[5] Lyons, James. "Welcome To Python_Speech_Features'S Documentation! — Python_Speech_Features 0.1.0 Documentation". Python-speech-features.readthedocs.org. N.p., 2013. Web. 18 Dec. 2015.

[6] Pal Singh, Parwinder. "An Approach To Extract Feature Using MFCC". IOSR Journal of Engineering 4.8 (2014): 21-25. Web.

[7] Reynolds, D.A., and R.C. Rose. "Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models". IEEE Transactions on Speech and Audio Processing 3.1 (1995): 72-83. Web.

[8] Reynolds, Douglas A., Thomas F. Quatieri, and Robert B. Dunn. "Speaker Verification Using Adapted Gaussian Mixture Models". Digital Signal Processing 10.1-3 (2000): 19-41. Web.