Experiments on Speech/Music Discrimination

Wei Jiang

wj2122@columbia.edu

Final Project – Digital Signal Processing ELEN E4810

1 Introduction

The problem of speech/music discrimination has become increasingly important as automatic speech recognition system are applied to more real-world multimedia domains. One of the issue in the design of a signal classifier is the selection of an appropriate feature set that captures the temporal and spectral structures of the signal. Many features have been used in speech/music discrimination. The cepstral coefficients and the posterior-based features are two widely used features. Another issue is the selection of a classification algorithm, and a popular approach to speech/music discrimination is to use the decorrelated feature frames to train distribution models to distinguish training data labelled as speech or music [4, 10]. In this report, we test the performance of three kinds of classic features (MFCCs, Dynamic features, and the posterior-based features) and two kinds of classic distribution models (GMM and K-nearest neighbor) in separating speech and music signals. Furthermore, we implement a demo to classify a given audio signal into speech or music category.

2 Feature Extraction

The speech wave is the response of the vocal tract to one or more excitation signals [5]. In spectral terms, the speech spectrum can be treated as the product of an excitation spectrum and a vocal tract spectrum. Let x denote the original speech signal, and let v and e denote excitation

source and the time-varying vocal tract respectively,

$$|X(e^{j\omega})| = |V(e^{j\omega})||E(e^{j\omega})| \quad (1)$$

$$\log|X(e^{j\omega})| = \log|V(e^{j\omega})| + \log|E(e^{j\omega})| \, . \quad (2)$$

For voiced sounds, E term usually corresponds to an event that is relative extended in time, and thus it vary relatively rapidly. While the V term varies more slowly. By performing a Fourier Transform on the logarithm of the spectrum, we can separate $\log|X(e^{j\omega})|$ into two log spectral components, one varies rapidly, and the other varies slowly with ω. This operation is called deconvolution. We call the DTFT of $\log|X(e^{j\omega})|$ as the cepstrum. And usually the cepstrum is given by IDTFT as:

$$c_n = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log|X(e^{j\omega})|e^{j\omega}d\omega \, , \quad (3)$$

where c_n is called the nth cepstral coefficient. The resonance spectral varies slowly with respect to frequency, and it yields large valued cepstral coefficients for low values of n. The spectral fine structure is more rapidly varying with ω, and it yields small-valued cesptral coefficients for small n. Thus selection of low-N cepstral separates resonance information from excitation sources. In practical, we take DFT instead of DTFT, which gives good result.

To capture the dynamics of the vocal tract movements, the short-term spectrum is typically computed every 10–20 ms using a window of 20–30 ms. One reason for computing the short-term spectrum is that the cochlea of the human ear performs a quasi-frequency analysis. The analysis in the cochlea takes place on a nonlinear frequency scale (the Bark scale or the mel scale). This scale is approximately linear up to about 1000 Hz and is approximately logarithmic thereafter [5]. So, in the feature extraction, it is very common to perform a frequency warping of the frequency axis after the spectral computation.

2.1 Mel-scaled cepstral coefficients

The mel-scaled cepstral coefficients (MFCCs) (plus an energy term) has been shown to be quite effective for speech recognition [4], and perhaps is the most popular features used for speech recognition today. The five-step process of extracting the MFCCs is illustrated in Fig.1 (refer to [5] for more details).

step 1: Pre-emphasize the spectrum to approximate the unequal sensitivity of human hearing at different frequencies. This step eliminate the effect of dc offsets in the speech signal. In

the experiment, this is done by passing the original audio signal through a high-pass filter: $H(z) = 1 - 0.97z^{-1}$

step 2: Compute the power spectral estimate for the analysis window. First the waveform is Hamming-windowed in overlapping steps. Each window is 25ms wide and are overlapped with 15ms width (the hop size between successive windows is 10 ms). Then for each window, the power spectrum is computed using a DFT and calculating its squared magnitude.

step 3: Integrate the log of the power spectrum within overlapping critical band filter responses. That is, the log spectral are perceptually weighted by a non-linear map of the frequency scale, which is known as the mel scale. The frequency scale is roughly linear below 1 kHz and roughly logarithmic above 1 kHz. The mel scale is based on pitch perception, which emphasizes mid-frequency bands in proportion to their perceptual importance.

step 4: Performing an IDFT. The mel-scaled spectrum is transformed into cepstral coefficients by using IDFT.

step 5: Performing liftering (filtering along frequency). The cepstral coefficients are multiplied by a simple function: n^{α} ($\alpha = 0.6$ in the experiments), where n is the cepstral index. This step modifies the distance that could be computed with these features to be more or less sensitive to the amplitude of resonant peaks in the spectrum.

With these steps, the original audio waveform is transformed into a sequence of 13-dimensional MFCC feature vector (12 cepstral coefficients plus an energy term).

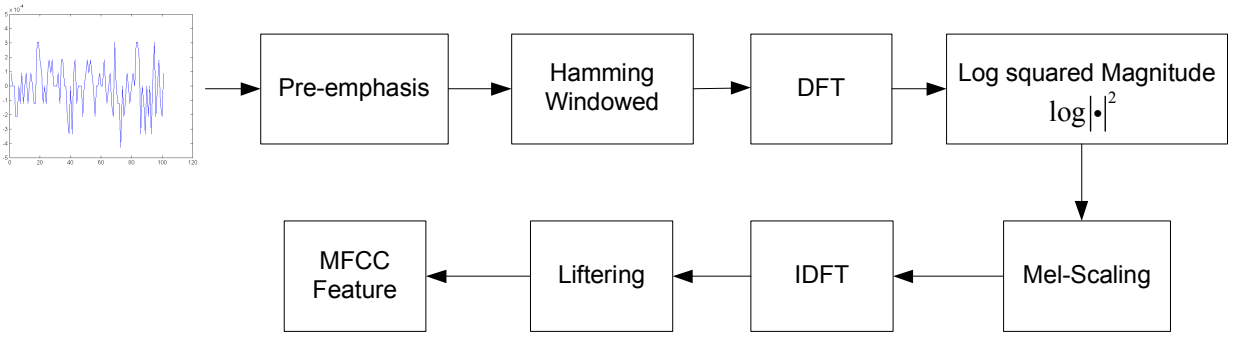

Figure 1: Example MFCC feature extraction.

2.2 Dynamic features

The MFCC feature provides good smooth estimates of local spectra. The local time derivatives of the cepstrum capture the dynamic behavior of speech [5]. The delta cepstrum is one of the most commonly used dynamic features in speech recognition. This is usually implemented as a least-squares approximation of the local slop:

$$\triangle c_i(n) = \frac{\sum_{k=-N}^{N} k c_i(n+k)}{\sum_{k=-N}^{N} k^2} , \tag{4}$$

where $c_i(n)$ is the nth cesptral coefficient of the ith frame. That is, each stream of delta cepstral values is computed by correlating the corresponding stream of cepstral values with a straight line that has a slop of one. The second derivative (the delta-delta cepstrum) is also often useful.

Note that since the delta cepstrum emphasizes the dynamic aspects of the speech spectrum over time, which is relatively insensitive to constant spectral characteristics, the dynamic features are usually combined with the cepstral features in a practical system. And in our experiment, we take $N = 2$, and use delta cepstrum and delta-delta cepstrum together as dynamic features.

2.3 Posterior-based features

State-of-the-art continuous speech recognition systems are statistical in nature, based on Hidden Markov Models (HMM) [1, 8]. Using HMM to model speech assumes that speech is a piecewise stationary process, that is, a utterance is modeled as a succession of discrete stationary states, with instantaneous transitions between these states. Fig.2 shows a simple example of Hidden Markov Modeling of a audio segment. f_j is the observation for the jth frame, which can be any one of the m observations $o_1, \ldots, o_m$. q_i is the true states of the ith frame, based on which the observation f_i is observed. q_i can be any one of the n states $s_1, \ldots, s_n$. Actually for each frame f_j, we can derive a set of posteriors $\{p(q_i^k|f_j)\}$, $q_i^k = s_k$, for $k = 1, \ldots, n$, based on the state transition matrix $\{p(s_j|s_i)\}$ and the observation matrix $p(o_i|s_i)$.

According to information theory, a channel designed for particular type of signal will exhibit characteristic behavior at its output when that signal is passed through the channel. In the case where the channel is an HMM trained to discriminate speech and music signals, it should therefore be possible to distinguish speech and music signals by examining the behavior of these probabilities [1]. In [1, 10], secondary posterior probability features (some statistics of the output of HMM) are derived from the hybrid systems. In our experiment, we test two kinds of posterior-based features: the average entropy and the average dynamism.

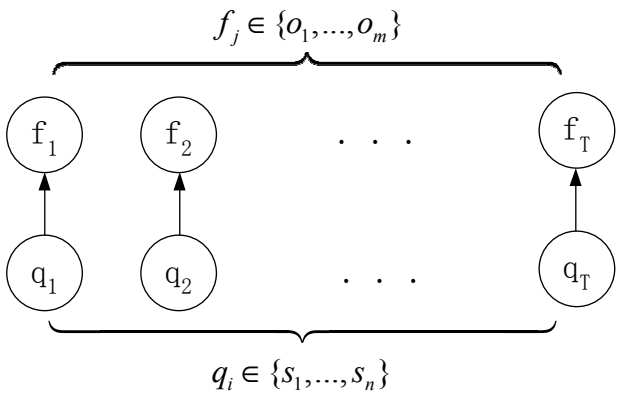

Figure 2: Example of Hidden Markov Modeling of audio segment. f_j is the observation for the jth frame, which can be any one of the m observations $o_1, \ldots, o_m$. q_i is the true state of the ith frame, based on which the observation f_i is observed. q_i can be any one of the n states $s_1, \ldots, s_n$. Actually for each frame f_j, we can derive a set of posteriors $\{p(q_i^k|f_j)\}$, $q_i^k = s_k$, for $k = 1, \ldots, n$, based on the state transition matrix $\{p(s_j|s_i)\}$ and the observation matrix $p(o_i|s_i)$.

Entropy is a measurement of the uncertainly in a given distribution. In the case of HMM outputs, the instantaneous entropy h_t at a specific time frame t is defined as:

$$h_t = -\sum_{k=1}^{n} P(q_t^k|f_t) \log P(q_t^k|f_t)$$

And the average entropy is defined as the average value of entropy over a window of several frames:

$$H_t = \frac{1}{N} \sum_{l=t-N/2}^{t+N/2} h_l \tag{5}$$

Dynamism is a measurement of the rate of change of a quantity. The instantaneous dynamism at time t is defined as:

$$d_t = \sum_{k=1}^{n} \left[P(q_t^k|f_t) - P(q_t^k|f_{t+1})\right]^2$$

This features captures the dynamic behavior of the probability values. And the average dynamism is given by:

$$D_t = \frac{1}{N} \sum_{l=t-N/2}^{t+N/2} d_l \tag{6}$$

The work flow of posterior-based feature extraction is as follows: First, in the training stage, we cluster each of the training set (speech and music training set), respectively, into 20 categories

(here we use Kmeans algorithm [3]). Each category corresponds to one observation o. Then we map the original audio segments into a sequence of observations, and train a HMM model for the speech and music category respectively based on the observation sequences. In the testing stage, given a testing signal, it is firstly converted into a observation sequence, and then is provided to the HMM models. The HMM models estimate, for speech category and music category respectively, a set of posteriors $\{p(q_i^k|f_j)\}$ for every frame f_j. Average entropy and average dynamism are calculated for each category The window size N is set to be 40.

3 Speech/Music Classification

In this report we consider the speech/music discrimination problem as a pure classification problem. That is, the training data are already segmented as pure speech or pure music signals. And also the test data are segmented speech or music signals. This kind of experiment is valuable to compare feature representation and classification algorithms in discriminating speech and music [9, 7]. In this section we test two classifiers, which are the MAP classifier with Gaussian Mixture Model, and the K-nearest neighbor classifier.

First we introduce some notations for the remaining part in this section. For a testing sound file $\mathbf{x}$, it is represented by $N \times D$ feature matrix – N frames, each represented by D-dimensional feature vectors. In our two-category classification problem, let ω_s and ω_m represent the state of nature of being speech and music category, respectively, and let y be the classification result for $\mathbf{x}$, where $y = 1$ if $\mathbf{x} \in \omega_s$, $y = 0$ if $\mathbf{x} \in \omega_m$.

3.1 MAP Classifier with Gaussian Mixture Model

We assume that the speech and the music classes both obey a Gaussian Mixture Model as follows:

$$\begin{aligned} p(\mathbf{x}|\theta_s) &= \sum_{i=1}^{n_s} p(z_{si}|\theta_s) p(\mathbf{x}|z_{si}, \theta_s) \\ p(\mathbf{x}|\theta_m) &= \sum_{i=1}^{n_m} p(z_{mi}|\theta_m) p(\mathbf{x}|z_{mi}, \theta_m) \ , \end{aligned}$$

where

$$\begin{aligned} p(\mathbf{x}|z_{si}, \theta_s) &= \frac{1}{(2\pi)^{d/2}|\Sigma_{si}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_{si})^T \Sigma_{si}^{-1}(\mathbf{x} - \mu_{si})\right\} \\ p(\mathbf{x}|z_{mi}, \theta_m) &= \frac{1}{(2\pi)^{d/2}|\Sigma_{mi}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_{mi})^T \Sigma_{mi}^{-1}(\mathbf{x} - \mu_{mi})\right\} \ , \end{aligned}$$

and $\theta_s = \{z_{si}, \mu_{si}, \Sigma_{si}, n_s\}$ and $\theta_m = \{z_{mi}, \mu_{mi}, \Sigma_{mi}, n_m\}$ are the parameter sets for speech class and music class respectively, where n_s and n_m are the number of Gaussian components in each classes, z_{si}, μ_{si}, Σ_{si} and z_{mi}, μ_{mi}, Σ_{mi} are, respectively the prior, mean vector and covariance matrix for each Gaussian component in speech and music classes. When n_s and n_m is given, Expectation Maximization (EM) method is used to learn the GMM for both classes [2], and the k-means algorithm is used to generate initial clusters for EM learning. In [6], an adaptive algorithm is developed to automatically determine the number of Gaussian components based on the *Minimum Message Length* (*MML*) criterion. In our experiments, we adopt the adaptive algorithm described in [6] to decide the number of Gaussian components and at the same time learn the GMM density.

When the GMM for both classes are learned, the MAP classifier can be used to classify a new coming signal into speech or music class as follows:

$$\begin{aligned} y = 1, &\quad p(\omega_s|\mathbf{x}) \geq p(\omega_m|\mathbf{x}) \\ y = 0, &\quad p(\omega_s|\mathbf{x}) < p(\omega_m|\mathbf{x}) \end{aligned}$$

3.2 K-Nearest Neighbor Classifier

The K-nearest neighbor classifier classifies a given testing data by assigning it the label most frequently represented among the k nearest training samples. That is, a decision is made by examining the labels on the k nearest neighbors and taking a vote. Usually k is odd to avoid ties [3]. For a testing sound file $\mathbf{x}$, let $\mathbf{x}_i$ be its ith frame, $\mathbf{x}_i$ is firstly classified as follows:

$$\begin{aligned} y_i = 1, &\quad \sum_{\tilde{\mathbf{x}}_i \in \mathcal{N}\{\mathbf{x}_i\}} I(\tilde{\mathbf{x}}_i \in \omega_s) \geq \tfrac{k}{2} \\ y_i = 0, &\quad \text{otherwise} \end{aligned} ,$$

where $\mathcal{N}\{\mathbf{x}_i\}$ is the k nearest neighbors in the training set. Then the decision rule for $\mathbf{x}$ will be:

$$\begin{aligned} y_i = 1, &\quad \tfrac{1}{N} \sum_{i}^{N} y_i \geq 0.5 \\ y_i = 0, &\quad \text{otherwise} \end{aligned} .$$

4 Implementation Issues

There are several issues needed to be considered when implementing the above algorithms.

- **Feature normalization**

The different types of features have a large diversity in their feature scales. This may cause big problem in classification. For example, when calculating the Euclidean distance between two frames, the feature vectors with large scales will dominate the distance. Thus it is necessary to first normalize different feature vectors into comparable scale before any further processing. In the experiment, the feature vectors are normalized as follows: for each audio segment $\mathbf{x}$,

$$\hat{x}_i = \frac{x_i - \mu(x_i)}{\sigma(x_i)}$$

where x_i is the ith feature element. $\mu(x_i)$ and $\sigma(x_i)$, respectively, are the mean and variance over all the frames of $\mathbf{x}$.

- **PCA to reduce the feature redundancy**

For the MAP classifier with GMM, a good estimation of the parameters in GMM is important to the classification performance. However, in our experiment, the training data is relatively insufficient to train a good GMM model with a large dimensionality. And we need to reduce the feature dimensionality first to reduce the redundancy in the original feature representation. The principle component analysis (PCA) method is used to this end.

- **Using prototypes for KNN classification**

In the classification process of K-nearest neighbor classification, we need to compare each frame in the testing data with every frame in every training data (the Euclidean distance is used here) to find the k nearest neighbors. This is very time consuming. To alleviate this problem, we first compress the training set by learning K_n representative prototypes for each category (sound and music), respectively. In such case, the testing frame is classified based on the representative prototypes only, which greatly reduces the computational cost. In the experiment, the K-means clustering algorithm is used to learn the prototypes, and K_n is set to be 20.

5 Experiments

5.1 Interface and programs for experiments

The user interface is shown in Fig.3, which comprises of mainly three parts: **Data Processing** (the top blue part), **Query Processing** (the middle pink part), and **Classification** (the bottom green part).

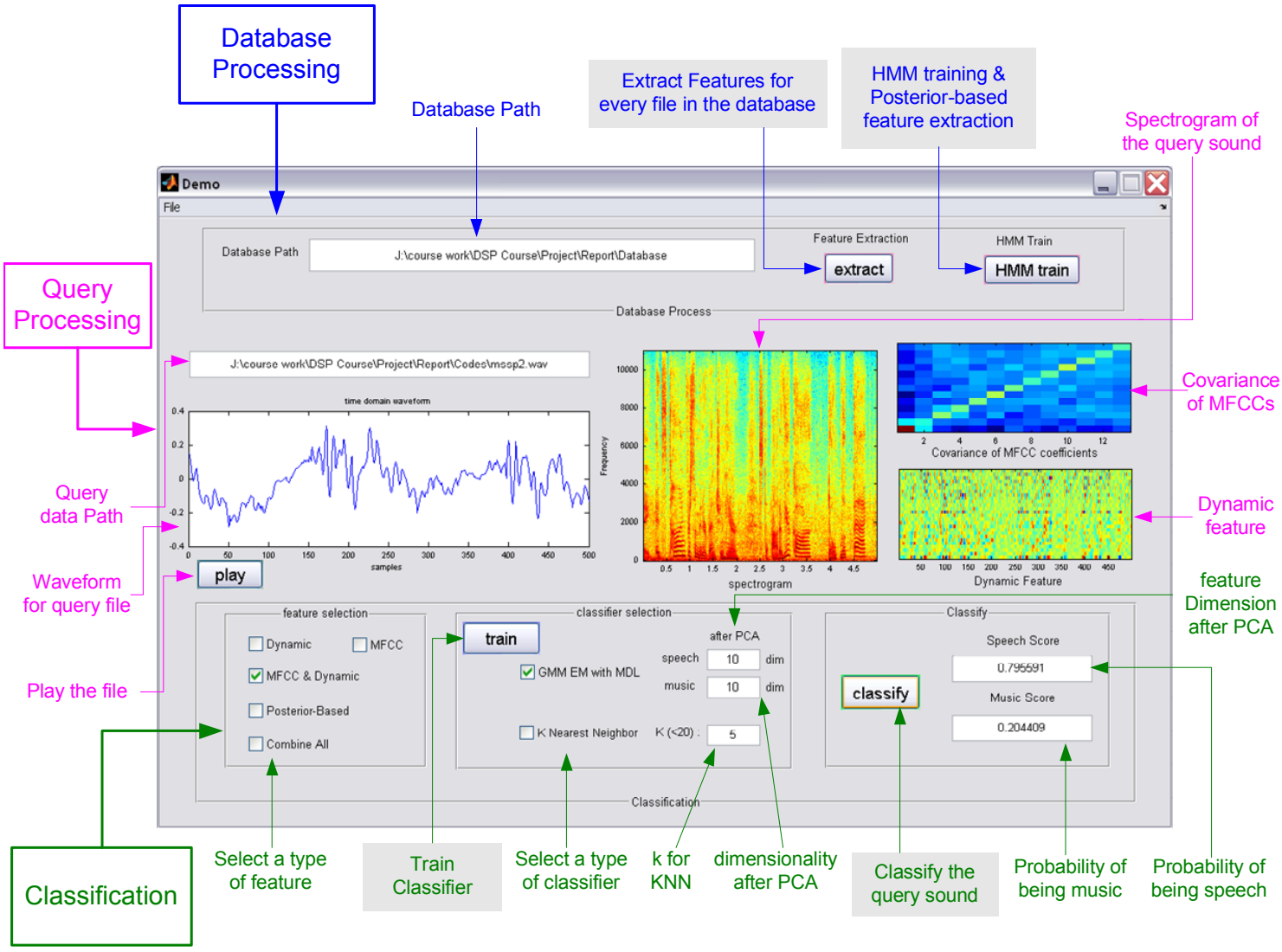

Figure 3: User interface

The programs are categorized into three categories as follows:

- Main programs for database pre-processing: feature extraction and prototype extraction
  - **BatchFeatureExtraction**: Extract feature vectors for all the sound files in the database.
  - **BatchPosteriorFeatureExtract**: Extract posterior-based for all the files in the database.
  - **FeatureExtraction**: Extract the feature vector for a single given sound file.
  - **SinglePosteriorFeatureExtract**: Extract the posetrior-baesd features for a single given sound file.
  - **melfcc**: Extract MFCC coefficients for a given sound waveform.
  - **deltas**: Calculate the deltas (derivatives) of a sequence.
  - **dhmm_em**: Learn the parameters of an HMM with discrete outputs using EM.
  - **dhmm_logprob**: Compute the log-likelihood of a dataset using a discrete HMM
  - **NormalizeFeature**: Normalize the feature vectors for all the sound files in the database.
- Main programs for classifier training
  - **BatchGMM**: Train GMM for positive and negative training sets respectively.
  - **GMMEM_Learn**: Learn GMM for a given data set with MDL model selection.
  - **BatchKNN**: Learn representative prototypes for positive and negative training sets respectively, which will be used for K-nearest neighbor classification, and will be used as the observation categories for HMM learning.
  - **BatchHMM**: Learn the Hidden Markov Models for the speech and music categories respectively.
- Main programs for classification
  - **GMMEMClassify**: MAP classification for a given testing data with GMM models.
  - **KNNClassify**: K-nearest neighbor classification for a given testing data.
  - **HMMClassify**: Generate the posteriors for a testing sound data.

Some of above programs are based on the open sources from the internet:

- **melfcc**: written by Prof. Ellis, 2005-04-19
  http://www.ee.columbia.edu/ dpwe/resources/matlab/rastamat/mfccs.html

– **deltas**: written by Prof. Ellis, 2003-06-30, dpwe@ee.columbia.edu

– **dhmm_em**: from Hidden Markov Model (HMM) Toolbox, written by Kevin Murphy (1998). http://www.ai.mit.edu/ murphyk/Software/hmm.html.

– **dhmm_logprob**: from Hidden Markov Model (HMM) Toolbox, written by Kevin Murphy (1998). http://www.ai.mit.edu/ murphyk/Software/hmm.html.

5.2 Experimental results

Two sets of experiments are carried out. First, we test the classification performance of different types of features and different types of classifiers. After that, we test the influence of some of the parameters on the classification performance.

- **Evaluation on classification performance**

The four music files (without vocal) – msmn1.wav, msmn2.wav msmn3.wav, msmn4.wav are used as training data for music category, and four speech files – mssp1.wav, mssp2.wav, mssp3.wav, mssp4.wav are used as training data for speech category. And all sound files are tested on the learned classifiers. The result is shown in Table.1 (a), (b), (c), (d) and (e), corresponding to the results of using MFCCs, dynamic features, combination of MFCCs and dynamic features, the posterior features, and the combination of all kinds of features, respectively. Here $k = 1$ for KNN classifier. That is, we test the nearest neighbor classifier.

From the tables we can see that, for MFCCs and dynamic features, in general the MFCCs has better performance than dynamic features. When using KNN classifier, incorporating dynamic features bring worse performance, while when using GMM classifier, combining dynamic features with MFCCs will improve the classification performance. Also in general, KNN (actually here is 1-NN) has better performance than GMM in discriminating speech and music. This may be caused by the insufficient data in estimating the parameters in GMM. We only have 1988 frames for each of the speech and music categories, while having a large dimensionality. Although PCA effectively reduces the feature redundancy, the data set may still be too small for estimating a good mixture model. When using posterior-based features, GMM outperforms KNN. In such case, the feature dimensionality is only 2, and the parameters in GMM may be well estimated. Thus the MAP classifier with GMM density estimation is expected to give better performance than KNN classifier. When all the features are combine together, the classification result is the best (both GMM and KNN correctly classify all the testing sound files in this case).

Table 1: Classification Results

(a) with MFCCs, $error_{KNN} = 0$, $error_{GMM} = 1/16$

test file	speech score		music score		labels	
	KNN	**GMM**	**KNN**	**GMM**	**KNN**	**GMM**
msmn1.wav	0.199195	0.277666	0.800805	0.722334	0	0
msmn2.wav	0.197183	0.350101	0.802817	0.649899	0	0
msmn3.wav	0.185111	0.231388	0.814889	0.768612	0	0
msmn4.wav	0.265594	0.334004	0.734406	0.665996	0	0
msmv1.wav	0.362173	0.531187	0.637827	0.468813	0	1
msmv2.wav	0.319920	0.336016	0.680080	0.663984	0	0
msmv3.wav	0.319920	0.338028	0.680080	0.661972	0	0
msmv4.wav	0.484909	0.416499	0.515091	0.583501	0	0
mssp1.wav	0.959759	0.869215	0.040241	0.130785	1	1
mssp2.wav	0.929577	0.742455	0.070423	0.257545	1	1
mssp3.wav	0.957746	0.851107	0.042254	0.148893	1	1
mssp4.wav	0.985915	0.905433	0.014085	0.094567	1	1
msms1.wav	0.841046	0.653924	0.158954	0.346076	1	1
msms2.wav	0.893360	0.830986	0.106640	0.169014	1	1
msms3.wav	0.710262	0.818913	0.289738	0.181087	1	1
msms4.wav	0.788732	0.623742	0.211268	0.376258	1	1

(b) with Dynamic Features, $error_{KNN} = 1/16$, $error_{GMM} = 6/16$

test file	speech score		music score		labels	
	KNN	**GMM**	**KNN**	**GMM**	**KNN**	**GMM**
msmn1.wav	0.241499	0.591594	0.758551	0.408457	0	1
msmn2.wav	0.273642	0.575453	0.726358	0.424547	0	1
msmn3.wav	0.179074	0.468813	0.820926	0.531187	0	0
msmn4.wav	0.287726	0.627761	0.712274	0.372233	0	1
msmv1.wav	0.480885	0.589537	0.519115	0.410463	0	1
msmv2.wav	0.309859	0.503018	0.690141	0.496982	0	1
msmv3.wav	0.356137	0.659960	0.643863	0.340040	0	1
msmv4.wav	0.311871	0.454728	0.688129	0.545272	0	0
mssp1.wav	0.975855	1.000000	0.024145	0.000000	1	1
mssp2.wav	0.923541	0.933602	0.076459	0.066398	1	1
mssp3.wav	0.935614	0.961771	0.064386	0.038229	1	1
mssp4.wav	0.969819	0.989940	0.030181	0.010060	1	1
msms1.wav	0.788732	0.903421	0.211268	0.096579	1	1
msms2.wav	0.929577	0.989940	0.070423	0.010060	1	1
msms3.wav	0.466801	0.635815	0.533199	0.364185	0	1
msms4.wav	0.857143	0.983903	0.142857	0.016097	1	1

(c) with MFCC & Dynamic Features, $error_{KNN} = 1/16$, $error_{GMM} = 0$

test file	**speech score**		**music score**		**labels**	
	KNN	**GMM**	**KNN**	**GMM**	**KNN**	**GMM**
msmn1.wav	0.215292	0.203219	0.784708	0.796781	0	0
msmn2.wav	0.229376	0.219316	0.770624	0.780684	0	0
msmn3.wav	0.189135	0.162978	0.810865	0.837022	0	0
msmn4.wav	0.217304	0.265594	0.782696	0.734406	0	0
msmv1.wav	0.336016	0.380282	0.663984	0.619718	0	0
msmv2.wav	0.344064	0.283702	0.655936	0.716298	0	0
msmv3.wav	0.323944	0.265594	0.676156	0.734406	0	0
msmv4.wav	0.519115	0.398390	0.480885	0.601610	1	0
mssp1.wav	0.939638	0.919517	0.060362	0.080483	1	1
mssp2.wav	0.925553	0.792757	0.074447	0.207243	1	1
mssp3.wav	0.925553	0.885312	0.074447	0.114688	1	1
mssp4.wav	0.965795	0.933602	0.034205	0.066398	1	1
msms1.wav	0.788732	0.694165	0.211268	0.305835	1	1
msms2.wav	0.881288	0.879276	0.118712	0.120724	1	1
msms3.wav	0.732394	0.696177	0.267606	0.303823	1	1
msms4.wav	0.796781	0.601610	0.203219	0.398390	1	1

(d) with Posterior Features, $error_{KNN} = 6/16$, $error_{GMM} = 3/16$

test file	**speech score**		**music score**		**labels**	
	KNN	**GMM**	**KNN**	**GMM**	**KNN**	**GMM**
msmn1.wav	0.501002	0.290581	0.498998	0.709419	1	0
msmn2.wav	0.735471	0.581162	0.264529	0.418838	1	1
msmn3.wav	0.703407	0.130261	0.296593	0.869739	1	0
msmn4.wav	0.368737	0.238477	0.631263	0.761523	0	0
msmv1.wav	0.625251	0.236473	0.374749	0.763527	1	0
msmv2.wav	0.545059	0.346693	0.454910	0.653307	1	0
msmv3.wav	0.404810	0.308617	0.595190	0.691383	0	0
msmv4.wav	0.412826	0.288577	0.587174	0.711423	0	0
mssp1.wav	0.430862	0.505010	0.569138	0.494990	0	1
mssp2.wav	0.547094	0.559118	0.452906	0.440882	1	1
mssp3.wav	0.679359	0.745491	0.320641	0.254509	1	1
mssp4.wav	0.733467	0.533066	0.266533	0.466934	1	1
msms1.wav	0.733467	0.583166	0.266533	0.416834	1	1
msms2.wav	0.735471	0.805611	0.264529	0.194389	1	1
msms3.wav	0.545090	0.494930	0.454910	0.535070	1	0
msms4.wav	0.557114	0.344689	0.422886	0.655311	1	0

(e) with all features combined together, $error_{KNN} = 0$, $error_{GMM} = 0$

test file	**speech score**		**music score**		**labels**	
	KNN	**GMM**	**KNN**	**GMM**	**KNN**	**GMM**
msmn1.wav	0.132265	0.248497	0.867735	0.751503	0	0
msmn2.wav	0.102204	0.262525	0.897796	0.737475	0	0
msmn3.wav	0.074148	0.200401	0.925852	0.799599	0	0
msmn4.wav	0.078156	0.372745	0.921844	0.627255	0	0
msmv1.wav	0.248497	0.420842	0.751503	0.579158	0	0
msmv2.wav	0.234469	0.292585	0.765531	0.707415	0	0
msmv3.wav	0.164329	0.266533	0.835671	0.733467	0	0
msmv4.wav	0.186373	0.462926	0.813627	0.537074	0	0
mssp1.wav	0.605201	0.921844	0.394790	0.078156	1	1
mssp2.wav	0.424850	0.707415	0.575150	0.292585	1	1
mssp3.wav	0.298597	0.837675	0.701403	0.162325	1	1
mssp4.wav	0.533066	0.925852	0.466934	0.074148	1	1
msms1.wav	0.332665	0.713427	0.667335	0.286573	1	1
msms2.wav	0.376754	0.867735	0.623246	0.132265	1	1
msms3.wav	0.701403	0.763527	0.298597	0.236473	1	1
msms4.wav	0.769539	0.641283	0.230461	0.358717	1	1

- **Evaluation on parameters**

There are several parameters that may influence the classification performance, including the number of the nearest neighbors k for KNN, and the window size and hop size in the extraction of MFCCs and dynamic features. Fig.4 shows the classification performance of KNN with different k, and with different window size (hop size is set to be half of the window size, i.e., the windows have 2:1 overlap). From the figure we can see that the nearest neighbor has the best performance, and as k increases, the classification result gets worse. Fig.5 shows the MAP classification result with GMM. From both Fig.4 and Fig.5, we see that KNN has better performance for small window size than for large window size, while GMM performs better for large window size. In general, when window size is 25ms both of the classifiers have fairly good performance. This phenomenon is consistent with the above discussion in Sec.2 that usually the window size is between 20-30 ms.

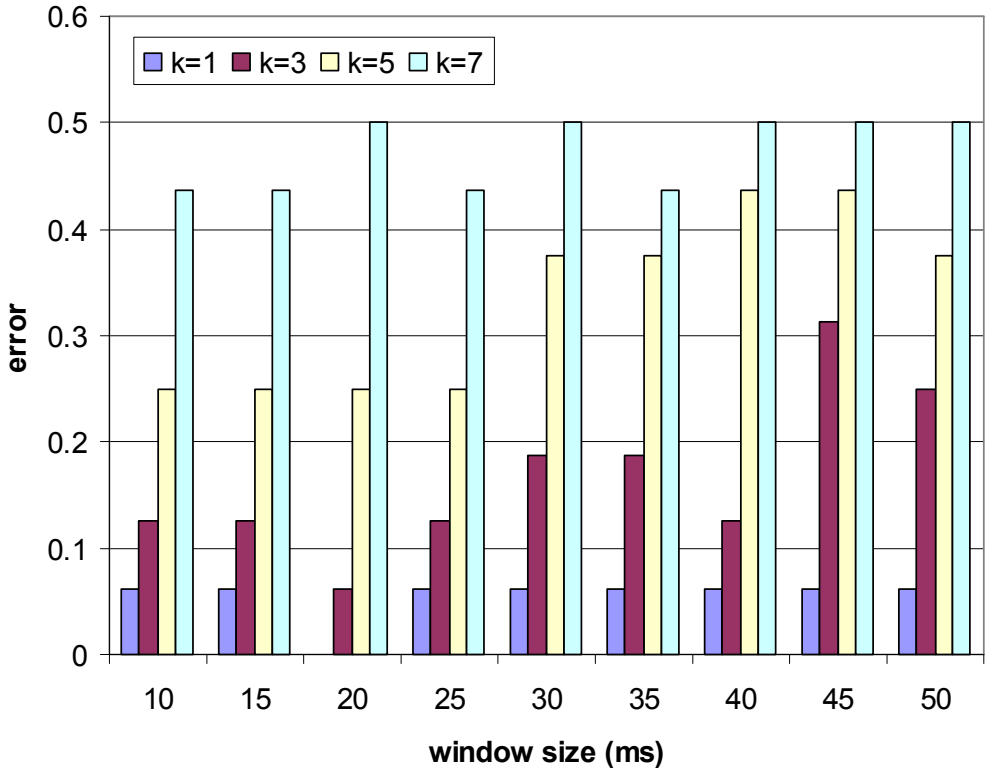

Figure 4: Classification performance of KNN with different k, and different window size and hop size.

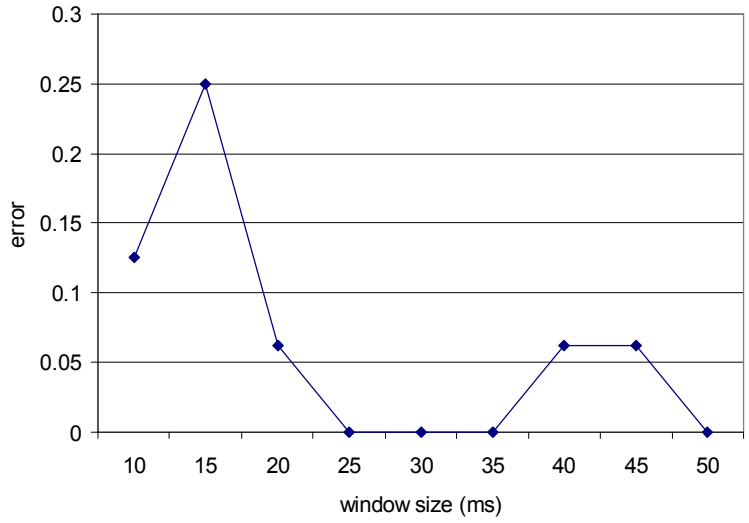

Figure 5: Classification performance of MAP classifier, using different window size and hop size.

6 Conclusions and Discussions

In this report, we take several experiments to test the performance of three kinds of classic features and two classic classifiers in discriminating music and speech. From the experiments, we can see that in general adding more features will improve the classification result. In the experiment, KNN outperforms MAP classifier with GMM in most cases, this may be cause by the difficulty to well estimate the parameters in GMM, since we only have 1988 training frames for each category while having large dimensionality for feature representation. Also, we find that the parameter k influences the performance of KNN classifier. In the experiments of this report, the nearest neighbor ($k = 1$) has the best performance. Furthermore, the window size and hop size in extracting MFCCs and dynamic feature vectors also influences the classification performance. When we decrease the window size to 15ms or increase the window size to 50ms, the performance seems to be no better than that of using 25ms window size and 10 ms hop size. This result is consistent with the discussion in Sec.2 that typically the window size is 20–30 ms, and the hop size is 10-20 ms.

The experiments in this report are based on the already well segmented speech/music segments. In practice, sound is encountered as a continuous stream that must be segmented as well as classified. Thus the speech/music discrimination system is usually designed to both segment and discriminate speech and music segments at the same time. Hidden Markov Model (HMM) is generally employed to this end, because it can take care of the dynamic behavior of audio signals. One classic approach is to use the hybrid HMM/ANN algorithm, with more sophisticate frameworks in both training and testing stage [1, 8] than what we did in this report. Discrimination between speech and music is really a difficult but very interesting topic. In the future, I will further explore this issue.

References

[1] J. Ajmera, I. McCowan, H. Bourlard, ”Speech/music segmentation using entropy and dynamism features in a HMM classification framework,” *Speech Communication*, 40(2003) 351-363.

[2] J. Bilmes, ”A gentle tutorial on the EM algorithm and its application to parameter estimation for gaussian mixture and hidden Markov models,” Technical Report ICSI-TR-97-021, International Computer Science Institute (ICSI), Berkeley, CA, 1997.

[3] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second edition, John Wiley and Sons, Inc., New York, USA, 2003.

[4] J.T. Foote, ”Content-based retrieval of music and audio,” *In C.C. J. Kuo et al., editor, Proc. of SPIE Multimedia Storage and Archiving Systems II*, Vol. 3229, pp. 138-147, 1997.

[5] B. Gold, N. Morgan, ”Speech and audio signal processing: processing and perception of speech and music,” John Wiley & Sons, Inc., USA, 1999.

[6] M.H.C. Law, M.A.T. Figueiredo, A.K. Jain, ”Simultaneous feature selection and clustering using mixture models,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(9) (2004) 1154-1166.

[7] Khaled El-Maleh, M. Klein, G. Petrucci, and P. Kabal, ”Speech/music discrimination for multimedia applications,” *Proc. IEEE Int. conf. on Accoustics, Speech, Signal Processing*, pp. 2445-2446, 2000.

[8] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco, ”Connectionist probability estimators in HMM speech recognition,” *IEEE Transactions on Speech and Audio Processing*, Vol. 2, No. 1, pp. 161-174, 1994.

[9] E. Scheirer, M. Slaney, ”Construction and evaluation of a robust multifeature speech/music discriminator,” *Proc. ICASSP*, pp. 1331-1334, Munich, Germany, 1997.

[10] G. Williams, D. Ellis, ”Speech/music discrimination based on posterior probability features,” *Proc. Eurospeech*, Budapest, September, 1999.