Deadlines for assignments

Date	Assignments	
Friday, November 11	1-3	
Monday, November 28	4-5	
Friday, December 9	6-8	undergrads
	6-10	grads

Each assignment needs to be emailed to Professor Meyer fmeyer@colorado.edu as a PDF file. Your file needs to have the following name: firstname_lastname.pdf. No other file format will be accepted.

PART 0: Introduction

The goal of this project is to explore some of the recent techniques developed in the audio industry to organize, and search large music collection by content. The explosion of digital music has created a need to invent new tools to search and organize music. Several websites provide large database of musical tracks:

- http://magnatune.com/
- http://www.allmusic.com/
- http://www.last.fm/

and also allow users to find musical tracks and artists that are similar.

Companies such as gracenote and shazam have application to recognize a song based solely on the actual music. Other examples of automated music analysis include

1. score following: Rock Prodigy, SmartMusic, Rockband
2. automatic music transcription: Zenph
3. music recommendation, playlisting: Google Music, Last.fm, Pandora, Spotify
4. machine listening: Echonest

At the moment these tools are still rudimentary. Fast computational methods are needed to expand these tools beyond their current primitive scope and integrate them on portable music players (e.g. iPod).

The goal of this project is to implement modern signal processing methods for content-based music information retrieval. In this project you will compute features to compare audio tracks, and to automatically detect the genre of a track. The features will be based on the human perception of music (psychoacoustic). These features can be computed using filterbanks, and lead to the definition of chroma.

In the first two parts of the project you will work on a small set of 12 music tracks. In the last part , you will test your algorithms on a a database of 150 training samples to learn the association between genres and songs. The performance of the algorithm will be evaluated using songs with know labels that are not part of the training set.

1.1 Terminology

For the purpose of this document we will be describing music using a terminology that is biased toward popular music. As a results we use the following words in a somewhat different context:

- an artist refers to the performer of a piece of music in the case of popular music, and a composer in the case of classical music. For instance, both Miles Davis and Johann Sebastian Bach are two artists.
- a song refers to a recording of (a part of) a piece of music. A song is identified with a CD track. The name does not imply any vocal performance.

1.2 The music

In the file http://ecee.colorado.edu/~fmeyer/.private/audio.zip you will find 12 tracks of various length. You will use these files to test the algorithms you develop in PARTS I & II. You will need the following information:

```
http://ecee.colorado.edu/~fmeyer/.private/audio.zip
username: music
password: eCmsHPaJeGT8GBue5WmAiUld
```

Please read the license posted on the webpage before downloading.

The 12 tracks are two examples of six different musical genres:

1. Classical
2. Electronic
3. Jazz
4. Metal and punk
5. Rock and pop
6. World

The name of the file indicates its genre. The musical tracks are chosen because they are diverse but also have interesting characteristics, which should be revealed by your analysis.

- track201-classical is a part of a violin concerto by J.S. Bach (13-BWV 1001 : IV. Presto).
- track204-classical is a classical piano piece composed by Robert Schumann (Davidsbundlertanze, Op.6. XVI).
- track370-electronic is an example of synthetic music generated a software that makes it possible to create notes that do not have the regular structure of Western music. The pitches may be unfamiliar but the rhythm is quite predictable.
- track396-electronic is an example of electronic dance floor piece. The monotony and the simplicity of the synthesizer and the bass drum are broken by vocals.
- track437-jazz is another track by the same trio as track439-jazz
- track439-jazz is an example of (funk) jazz with a Hammond B3 organ, a bass, and a percussion. The song is characterized by a well defined rhythm and a simple melody.
- track463-metal is an example of rock and metal with vocals, bass, electric guitars and percussion.
- track492-metal is an example of heavy metal with heavily distorted guitars, drums with double bass drum.
- track547-rock is an example of new wave rock of the 80's. The music varies from edgy to hard. It has guitars, keyboards, and percussion.
- track550-rock is an example of pop with keyboard, guitar, bass, and vocals. There is a rich melody and the sound of steel-string guitar.
- track707-world: this is a Japanese flute, monophonic, with background sounds between the notes. The notes are held for a long time.
- track729-world: this is a piece with a mix of Eastern and Western influence using electric and acoustic sarod and on classical steel-string guitars.

PART I: From the wav file to the psychoacoustic features

Sounds are produced by the vibration of air; sound waves produce variations of air pressure. The sound waves can be measured using a microphone that converts the mechanical energy into electrical energy. Precisely, the microphone converts air pressure into voltage levels. The electrical signal is then sampled in time at a sampling frequency f_s, and quantized. The current standard for high quality audio and DVD is a sampling frequency of $f_s = 96$kHz, and a 24-bit depth for the quantization. The CD quality is 16 bit sampled at $f_s = 44.1$ kHz.

2.1 Frames, and Samples

In order to automatically analyze the music, the audio file is divided into non overlapping intervals of a few milliseconds over which the analysis is conducted. In this project the audio files are sampled at $f_s = 11,025$ or $22,050$ Hz, and we consider intervals of size $N = 512$ samples. This interval of 46, or 23, millisecond is called a **frame**. Throughout the project we will always split a track into non overlapping frames. The number of frames is denoted by N_F.

Music is made up of notes of different pitch. Our goal is the reconstruction of a musical score. It is only natural that most of the automated analysis of music should be performed in the spectral (frequency) domain. Our analysis requires $N = 512$ samples to compute notes over a frame. The spectral analysis proceeds as follows.

Each frame is smoothly extracted by multiplying the original audio signal by a tapper window w. The Fourier transform of the windowed signal is then computed. If x_n denotes a frame of size $N = 512$ extracted at frame n, and w is a window of size N, then the Fourier transform, X_n (of size N) for the frame n is given by

```
Y  = fft (w.*xn);
K  = N/2 + 1;
Xn = Y(1:K);
```

2.2 Basic Psychoacoustic Quantities

In order to develop sophisticated algorithms to analyze music based on its content, we need to define several subjective features such as timbre, melody, harmony, rhythm, tempo, mood, lyrics, etc. Some of these concepts can be defined formally, while others are more subjective and can be formalized using a wide variety of different algorithms. We will focus on the features that can be defined mathematically.

2.3 Psychoacoustic

Psychoacoustics involves the study of the human auditory system, and the formal quantification of the relationships between the physics of sounds, and our perception of audio. We will describe some key aspects of the human auditory system:

1. the perception of frequencies and pitch for pure and complex tones;
2. the frequency selectivity of the auditory system: our ability to perceive two similar frequencies as distinct;

3. the modeling of the auditory system as a bank of auditory filters;
4. the perception of loudness;
5. the perception of rhythm.

2.4 Perception of frequencies

The auditory system, like the visual system, is able to detect frequencies over a wide range of scales. In order to measure frequencies over a very large range, it operates using a logarithmic scale. Let us a consider a pure tone, modeled by a sinusoidal signal oscillating at a frequency ω. If $\omega < 500$ Hz, then the perceived tone – or pitch – varies as a linear function of ω. When $\omega > 1,000$ Hz, then the perceived pitch increases logarithmically with ω.

Several frequency scales have been proposed to capture the logarithmic scaling of frequency perception.

2.5 The mel/Bark scale

The Bark (named after the German physicist Barkhausen) is defined as

$$z = 7\text{arcsinh}\left(\omega/650\right) = 7\log\left(x/650 + \sqrt{1 + (x/650)^2}\right),$$

where ω is measured in Hz. The mel-scale is defined by the fact that 1 bark = 100 mel. In this project we will use a slightly modified version of the mel scale defined by

$$m = 1127.01048 * \log\left(1 + \omega/700\right). \tag{1}$$

2.6 The cochlear filterbank

Finally, we need to account for the fact that the auditory system behaves as a set of filterbanks, with overlapping frequency responses. For each filter, the range of frequencies over which the filter response is significant is called the critical band. Our perception of pitch can be quantified using the total energy at the output of each filter bank. All spectral energy that falls into one critical band is summed up, leading to a single number for that frequency band.

We describe in the following a simple model of the cochlear filterbank. The filter bank is constructed using $N_B = 40$ logarithmically spaced triangle filters centered at the frequencies Ω_p, defined by

$$\text{mel}_n = 1127.01048 * \log\left(1 + \Omega_p/700\right), \tag{2}$$

where the sequence of mel frequencies is equally spaced in the mel scale,

$$\text{mel}_n = n\frac{\text{mel}_{\text{max}} - \text{mel}_{\text{min}}}{N_B}, \tag{3}$$

with

$$\text{mel}_{\text{max}} = 1127.01048 * \log\left(1 + 0.5 * \omega_s/700\right), \tag{4}$$

$$\text{mel}_{\text{min}} = 1127.01048 * \log\left(1 + 20/700\right), \tag{5}$$

and $N_B = 40$. Each filter H_p is centered around the frequency Ω_p, and defined by

$$H_p(\omega) = \begin{cases} \dfrac{2}{\Omega_{p+1} - \Omega_{p-1}} \dfrac{\omega - \Omega_{p-1}}{\Omega_p - \Omega_{p-1}} & \text{if} \quad \omega \in [\Omega_{p-1}, \Omega_p), \\ \dfrac{2}{\Omega_{p+1} - \Omega_{p-1}} \dfrac{\Omega_{p+1} - \omega}{\Omega_{p+1} - \Omega_p} & \text{if} \quad \omega \in [\Omega_p, \Omega_{p+1}). \end{cases} \tag{6}$$

Each triangular filter is normalized such that the integral of each filter is 1. In addition, the filters overlap so that the frequency at which the filter H_p is maximum is starting frequency for the next filter h_{n+1}, and the edge frequency of h_{n-1}.

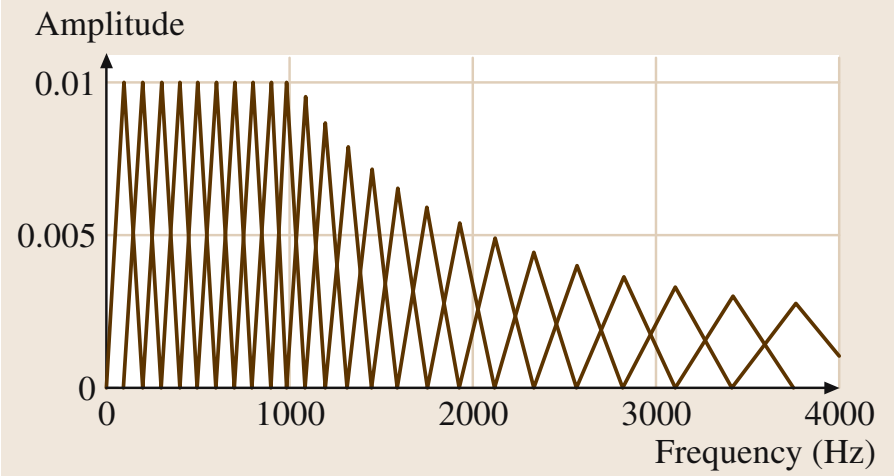

Figure 1: The filterbanks used to compute the mfcc.

Finally, the mel-spectrum (MFCC) coefficient of the n-th frame is defined for $p = 1, \ldots, N_B$ as

$$\texttt{mfcc}[p] = \sum_{k=1}^{K} |H_p(k) X_n(k)|^2 \tag{7}$$

where the filter H_p is a discrete implementation of the continuous filter defined by (6). The discrete filter is normalized such that

$$\sum_j H_p(j) = 1, p = 1, \ldots, N_B. \tag{8}$$

The MATLAB code in the following next three pages computes the `mfcc` coefficients. The computation of filter banks H_p is already completed. You need to implement (7) in the Fourier domain.

```
function [mfcc] = mfcc(wav, fs, fftSize, window)
%
% USAGE
%   [mfcc] = mfcc(wav, fs, fftSize,window)
%
% INPUT
%      vector of wav samples
%      fs : sampling frequency
%      fftSize: size of fft
%      window: a window of size fftSize
%
% OUTPUT
%   mfcc (matrix) size coefficients x nFrames


%   hardwired parameters

hopSize = fftSize/2;
nBanks = 40;

%    minimum and maximum frequencies for the analysis
fMin = 20;
fMax = fs/2;

%_____________________________________________________________________
%
%     PART 1 : construction of the filters in the frequency domain
%_____________________________________________________________________

%  generate the linear frequency scale of equally spaced frequencies from 0 to fs/2.

linearFreq = linspace(0,fs/2,hopSize+1);

fRange = fMin:fMax;

%  map the linear frequency scale of equally spaced frequencies from 0 to fs/2
% to an unequally spaced mel scale.

melRange = log(1+fRange/700)*1127.01048;
```

```
% The goal of the next coming lines is to resample the mel scale to create uniformly
% spaced mel frequency bins, and then map this equally spaced mel scale to the linear
& scale.

% divide the mel frequency range in equal bins

melEqui = linspace (1,max(melRange),nBanks+2);

fIndex = zeros(nBanks+2,1);

% for each mel frequency on the equally spaces grid, find the closest frequency on the
% unequally spaced mel scale

for i=1:nBanks+2,
    [dummy fIndex(i)] = min(abs(melRange - melEqui(i)));
end

% Now, we have the indices of the equally-spaced mel scale that match the unequally-spaced
% mel grid. These indices match the linear frequency, so we can assign a linear frequency
% for each equally-spaced mel frequency

fEquiMel = fRange(fIndex);

%  Finally, we design of the hat filters. We build two arrays that correspond to the center,
%  left and right ends of each triangle.

fLeft   = fEquiMel(1:nBanks);
fCentre = fEquiMel(2:nBanks+1);
fRight  = fEquiMel(3:nBanks+2);

% clip filters that leak beyond the Nyquist frequency

[dummy, tmp.idx] = max(find(fCentre <= fs/2));
nBanks = min(tmp.idx,nBanks);

% this array contains the frequency response of the nBanks hat filters.

freqResponse = zeros(nBanks,fftSize/2+1);

hatHeight = 2./(fRight-fLeft);
```

```
% for each filter, we build the left and right edge of the hat.

for i=1:nBanks,
    freqResponse(i,:) = ...
        (linearFreq > fLeft(i) & linearFreq <= fCentre(i)).* ...
        hatHeight(i).*(linearFreq-fLeft(i))/(fCentre(i)-fLeft(i)) + ...
        (linearFreq > fCentre(i) & linearFreq < fRight(i)).* ...
        hatHeight(i).*(fRight(i)-linearFreq)/(fRight(i)-fCentre(i));
end

%
% plot a pretty figure of the frequency response of the filters.

figure;set(gca,'fontsize',14);semilogx(linearFreq,freqResponse');
axis([0 fRight(nBanks) 0 max(freqResponse(:))]);title('FilterbankS');

%_____________________________________________________________
%
% PART 2 : processing of the audio vector In the Fourier domain.
%_____________________________________________________________
%
%  YOU NEED TO ADD YOUR CODE HERE
```

Assignment

1. Write a MATLAB function that extract T seconds of music from a given track. You will use the MATLAB function `audioread` to read a track and the function `play` to listen to the track.
2. Implement the computation of the mfcc coefficients, as defined in (7). You simply need to add your code in the MATLAB code in the previous pages.
3. Evaluate your MATLAB function `mfcc` on the 12 audio tracks, and display the output as an image using `imagesc`. You will use T =**24 seconds** from the middle of each track and compute a matrix of `mfcc` coefficients of size $N_B = 40$ rows and $24 \times 11,025/512 = 517$ columns.

PART II: Tonality and Chroma

3.1 The equal-tempered scale

We now investigate the time-frequency structure related to the concept of tonality and chroma. The tonality is concerned with the distribution of notes at a given time. In contrast, the melody characterizes the variations of the spectrum as a function of time.

We first observe that the auditory sensation that is closely related to frequency, also known as pitch, corresponds to a logarithmic scale of the physical frequency. We have seen examples of such scales in PART I: the bark and the mel scales.

In this PART, we define another logarithmic scale, known as the *tempered scale*. First, we define a reference frequency f_0 associated with a reference pitch. While it is customary to use the frequency 440 Hz associated with the pitch A4, we will also use $f_0 = 27.5Hz$ (this known as A0, the lowest note on a piano) . The tempered scale introduces 12 frequency intervals – known as semi-tones – between this reference frequency f_0 and the next frequency also perceived as an A, $2f_0$. As a result, a frequency f_{sm} that is sm semitones away from f_0 is given by

$$f_{sm} = f_0 2^{sm/12}. \tag{9}$$

An interval of 12 semitones, $[f_{sm}, f_{sm+12})$, corresponds to an octave. The same notes (e.g. A, or C#) are always separated by an octave. For instance, the two previous As, A0 and A4 are separated by 4 octaves, since $440 = 2^4 \times 27.5$. The notion of note can be formally modeled by the concept of chroma.

3.2 Chroma

The chroma is associated with the relative position of a note inside an octave. It is a relative measurement that is independent of the absolute pitch. We describe an algorithm to compute a chroma feature vector for a frame n.

To lighten the notation, we drop the dependency on the frame index n in this discussion.

We first present the idea informally, and then we make our statement precise. We are interested to map a given frequency f to a note. Given a reference frequency f_0, then we can use (9) to compute the distance sm between f and f_0 measured in semitones,

$$sm = \text{round}(12 \log_2(f/f_0)). \tag{10}$$

We note that f is usually not exactly equal to $f_0 2^{sm/12}$, and this is why we round $12 \log_2(f/f_0)$ to the closest integer.

We can then map this index sm to a note, or chroma, c within an octave, using

$$c = sm \mod (12), \tag{11}$$

or

$$sm = 12q + c, \quad \text{with} \quad 0 \leq c \leq 11, \quad \text{and} \quad q \in \mathbb{N}. \tag{12}$$

In other words, f_{sm} and f_{sm+12} given by (9) correspond to the same note c, which is exactly what you see if you look at the keys of a piano.

The problem with this approach is that we do not have an algorithm to extract the frequencies f of the notes that are being played at a given time. Rather, we obtain a distribution of the spectral energy provided by

the Fourier transform. We need to identify the main peaks in the Fourier transform, and compute the chroma associated with these frequencies, using the equations above.

In the following we describe the different steps of the algorithm in details.

Step 1: spectral analysis and peak detection

The goal is to identify the main notes being played , and remove the noise and the harmonics coming from the timbre of each instrument.

For each frame n, we compute the windowed FFT as explained in PART I. We obtain $K = N/2 + 1$ (where N is even) Fourier coefficients. We detect the local maxima of the magnitude of the Fourier transform $|X_n|$, and count the number of maxima, or peaks, n_{peaks}. We denote by $f_k, k = 1, \ldots, n_{\text{peaks}}$ these peak frequencies.

Step 2: Assignment of the peak frequencies to semitones

For each peak frequency f_k, we find the semitone sm and the corresponding frequency $f_{sm} = f_0 2^{sm/12}$ closest to f_k, so that

$$sm = \text{round}(12 \log_2(f_k/f_0)). \tag{13}$$

Finally, we map the index sm to the note c defined by

$$c = sm \mod (12). \tag{14}$$

For computational efficiency we use $f_0 = 27.5$ Hz instead of 440 Hz.

Step 3: the Pitch Class Profile: a weighted sum of the semitones

Instead of assigning each peak frequency $f_k$to a single semitone sm, we spread the effect of f_k to the two neighboring semitones sm and $sm + 1$ using a raised cosine function, defined in a weight function $w(k, c)$.

We define the *Pitch Class Profile* associated with the note, or chroma $c = 0, \ldots, 11$ as the weighted sum of all the peak frequencies that are mapped to the note c, irrespective of the octave they fall in,

$$PCP(c) = \sum_k w(k, c)|X_n(k)|^2, \quad \text{where } k \text{ is such that } c = \text{round}(12 \log_2(f_k/f_0)) \text{mod } 12, \tag{15}$$

and the weight $w(k, c)$ is given by

$$w(k, c) = \begin{cases} \cos^2(\pi\ r/2) & \text{if } -1 < r = 12 \log_2(f_k/f_0) - sm < 1, \\ & \text{where } c = sm \mod (12), \text{ and } sm = \text{round}(12 \log_2(f_k/f_0)), \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

In plain English, $PCP(c)$ tells us how much energy is associated with frequencies that are very close (as defined by the weight $w(k, c)$) to the note c across all the octaves.

Step 4: Normalize the Pitch Class Profile

We want our definition of chroma to be independent of the loudness of the music. We therefore need to normalize each PCP by the total PCP computed over all the semitones. The normalized pitch class profile (NPCP) is defined by

$$NPCP(c) = \frac{PCP(c)}{\sum_{q=1}^{12} PCP(q)} \tag{17}$$

Assignment

4. Implement the computation of the Normalized Pitch Class Profile, defined by (17). You will compute a vector of 12 entries for each frame n.

5. Evaluate and plot the NPCP for the 12 audio tracks found in
   http://ecee.colorado.edu/~fmeyer/.private/audio.zip
   See Fig. 2 for an example.

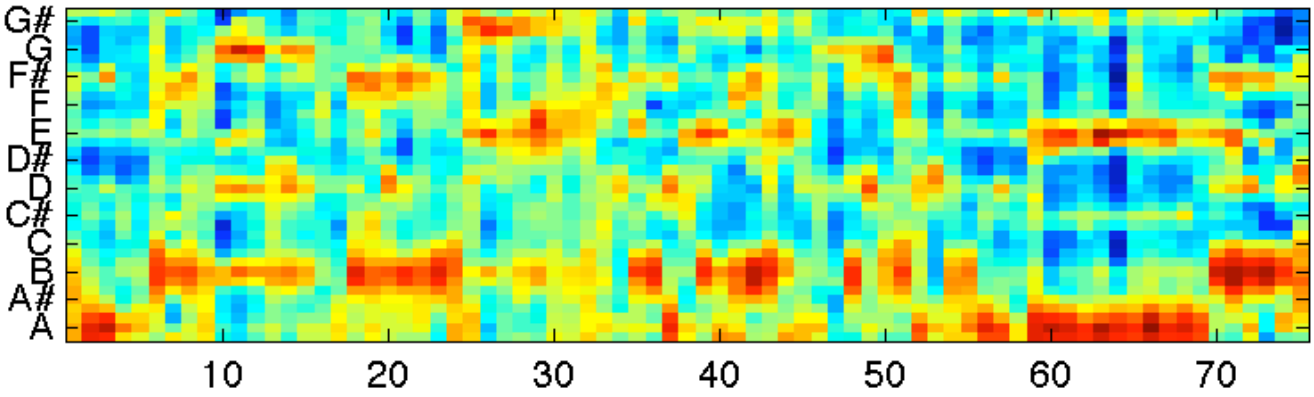

Figure 2: Chroma as a function of time (seconds).

PART III: Genre classification

While the definition of a musical genre is clearly arbitrary, and influenced by a culture, we will define genre classification as the problem of classifying music according to a set of (personal) labels. In particular, we will consider the following labels:

1. classical (western classical music)
2. electronic
3. jazz/blues
4. metal/punk
5. pop/rock
6. world.

The last label "world" is really a category that is only defined as being the complement of the union of the other categories. We note that you are not asked to refine the classification beyond the six categories.

The goal of PART III is to develop an algorithm that can upload a song and return a label with a certain probability. For instance, if the song "Recitative" from the album "Dark Intervals" by Keith Jarret is selected, one would expect that the algorithm classifies it as "classical" or "jazz" with a high probability. The artist Keith Jarrett is a clear example of the difficulty of such a genre classification, and the need for a probabilistic framework. The lab focuses on the genre classification question.

4.1 The data

4.2 Raw audio data

The data available are Pulse Coded Modulation (PCM) encode audio files. These format provides a digital sampling of the acoustic wave created by the sound pressures changes as a function of time. The audio has the following characteristics:

- sampling rate: 22,050 Hz
- sample size: 16 bit
- Number of channels: 1 (mono)
- Encoding: WAV

4.3 Available files

The composition of the training dataset is as follows:

- classical: 25 songs
- electronic: 25 songs

- jazz/blues: 25 songs
- metal/punk: 25 songs
- rock/pop: 25 songs
- world: 25 songs

Each song is a track of a CD and last for several minutes. While songs may have different lengths, we will work with an excerpt of fixed length extracted from the centre of the track.

4.4 But where are the songs?

A 1.4 GB archive with the 150 songs is available here:
http://ecee.colorado.edu/~fmeyer/.private/contest.zip

4.5 Copyright

These songs are made available to you for educational purposes only. It is illegal to use these songs for commercial purposes. These tracks are made available under the non-commercial use as defined by the Creative Commons License: http://creativecommons.org/licenses/by-nc-sa/1.0/legalcode.

4.6 Features

We extract an audio excerpt of 2 minutes from the center of each piece, and compute several audio features, and combine them into a vector $\boldsymbol{X}$. If the track is shorter than 2 minutes, we use the entire track. We will work with frames of size 512 samples (23 ms for the sampling rate equal to 22,050 Hz), and an overlap between frames of 50% = 256 samples. We have $N_F = 10,335$ overlapping frames of size 512 in 2 minutes of music.

We consider the following two vectors of features,

1. MFCC coefficients,
2. chroma (Normalized Pitch Class Profile)

Both sets of features yield a matrix $\boldsymbol{X}(k, n)$ that depends on a frequency (or pitch) index $k = 1, \ldots, K$, and a frame index $n = 1, \ldots, N_F$. We can think informally about $\boldsymbol{X}(1 : K, n)$ as the set of notes being played at time n.

In order to obtain more reliable classification results, and speed up the computation, we merge some of the Mel banks. We retain only 12 bands, as in the chroma representation. The new bands are defined by the next lines of MATLAB code.

```
t = zeros(1,36); (2.21)
t(1) =1;t(7:8)=5;t(15:18)= 9;
t(2)   = 2; t( 9:10) = 6; t(19:23) = 10;
t(3:4) = 3; t(11:12) = 7; t(24:29) = 11;
t(5:6) = 4; t(13:14) = 8; t(30:36) = 12;
```

```
mel2 = zeros(12,size(mfcc,2));

for i=1:12,
    mel2(i,:) = sum(mfcc(t==i,:),1);
end
```

In the remaining, we will consider that the mfcc coefficients are the 12 merged mfcc coefficients; in other words

```
mfcc = mel2;
```

4.7 Distance between features

Given two tracks s_1 and s_2, we are interested in comparing the distribution of notes between the two tracks. We proceed as follows.

For each track, the distribution of vectors $\boldsymbol{X}(:,n), n = 1,\ldots,N_F$ is modeled as a multivariate Gaussian distribution in $\mathbb{R}^K$, where $K = 12$. In essence, this approach collapses all the frames together, and summarizes the 2-minute excerpt by a mean note, and a covariance matrix. In MATLAB, we compute

```
>> mu = mean(mfcc,2);
>> Cov = cov(mfcc');
```

To compare two tracks, we compare the distance between the two Gaussian distributions $G^1 = \mathcal{N}(\mu_1, \Sigma_1)$ and $G^2 = \mathcal{N}(\mu_2, \Sigma_2)$ that are estimated for each track. We use a standard approach that is used in statistics, and we compute a symmetric version of the Kullback-Leibler divergence

$$KL(G^1, G^2) = \frac{1}{2}\mathrm{tr}(\Sigma_2^{-1}\Sigma_1 + \Sigma_1^{-1}\Sigma_2) + \frac{1}{2}\mathrm{tr}(\mu_1 - \mu_2)^T\left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)(\mu_1 - \mu_2) - K \tag{18}$$

Finally, the KL distance is rescaled using an exponential kernel, and we define the distance between the tracks s_1 and s_2 as

$$d(s_1, s_2) = \exp\left(-\gamma KL(G^1, G^2)\right), \tag{19}$$

where the parameter γ is chosen in $[0, 1]$ to optimize the classification. The

The MATLAB code in Fig. 3 computes the distance d between the two tracks `s1` and `s2`.

```
iCo1 = inv(Co1);
iCo2 = inv(Co2);
KL   =  0.5*(trace(Co1*iCo2) + trace(Co2*iCo1) + trace((iCo1+iCo2)*(m1-m2)*(m1-m2)');
gam  = 0.5;
d    = -exp(-KL/gam).
```

Figure 3: Distance d between the two tracks `s1` and `s2` .

4.8 Distance matrix

Assignment

6. Implement the computation of the distance d given by (19), and Fig. 3. Your function should be able to use the 12 merged MFCC coefficients or the Normalized Pitch Class Profile.

7. Compute the matrix of pairwise distance

$$D(s_1, s_2), s, s_2 = 1, \ldots, 150. \tag{20}$$

   Display the distance matrix as an image, and discuss its structure.

8. Compute the 6×6 average distance matrix between the genres, defined by

$$\overline{D}(i, j) = \frac{1}{25^2} \sum_{s_1 \in \text{genre } i,\ s_2 \in \text{genre } j} d(s_1, s_2), \quad i, j = 1, \ldots, 6, \tag{21}$$

   Experiment with different values of γ, and find a value of γ that maximizes the separation between the different genres, as defined by the 6×6 average distance matrix $\overline{D}$.

4.9 Classification method

We are now equipped with a distance to measure the similarity between two tracks. We describe here a simple procedure to classify a song with unknown genre using the training data.

K-Nearest Neighbors

Given a song s, we wish to determine its genre. We proceed as follows.

We compute the distance between the song s and every other song in the training data, and we determine the five nearest songs (according to d defined by (19)). Among the five nearest songs, we find the genre that is the most represented, and assign this genre to the track s.

Assignment

9. Implement a classifier based on the following ingredients, as explained above,
   - computation of the 12 mfcc coefficients, or 12 Normalized Pitch Class Profile
   - modified Kullback-Leibler distance d defined by (19)
   - genre = majority vote among the 5 nearest neighbors
10. Using cross validation, as explained in section 4.12, evaluate your classification algorithm. You will compute the mean and standard deviation for all the entries in the confusion matrices.

4.10 Statistical Evaluation of the Performance of My Algorithm

4.11 Confusion matrix

The accuracy of the classification will be quantified using confusion matrices. In this lab, a confusion matrix will measure the correct classification rate of each genre. For a given classification experiment, you will construct a 6×6 matrix where the rows are the true genres, and the columns are the genres classified by your algorithm. The entry $R(i, j)$ of the confusion matrix R is the number of songs of genre i that were classified as genre j. In the example in Table 1 the total number of songs per class was:

$$\begin{bmatrix} 21 & 3 & 1 & 0 & 0 & 0 \end{bmatrix},$$

so there were 21 good identifications (diagonal entries of the matrix) out of 25 queries. This corresponds to 84 % correct answers overall. The correct classification rate per class was:

$$\begin{bmatrix} 0.84 & 0.88 & 0.52 & 0.92 & 0.36 & 0.40 \end{bmatrix}, \tag{22}$$

which yields a 51.48% correct classification if we normalize with respect to the probability of each class.

	classical	electronic	jazz	punk	rock	world
classical	21	3	1	0	0	0
electronics	3	22	0	0	0	0
jazz	0	1	13	2	8	1
punk	0	0	0	23	1	1
rock	0	2	3	6	9	5
world	3	7	1	0	4	10

Table 1: Example of a confusion matrix. The rows are the true genres, and the columns are the predicted genres.

4.12 Cross-validation

In order to evaluate the performance of your method, you will perform a 5-fold cross-validation 10 times. Each cross-validation experiment consists in the following sequence of operations.

- let $\mathcal{S}$ be the set of all 150 songs.
- let $\mathcal{G}_i, i = 1, \cdots, 6$ be the six sets of songs organized by genres:

$$\mathcal{S} = \bigcup_{i=1}^{6} \mathcal{G}_i.$$

- **repeat n =1:10** // `average over the randomization`

1. randomly divide each genre $\mathcal{G}_i$ into 5 subsets of size 5

$$\mathcal{G}_i = \bigcup_{k=1}^{5} \mathcal{G}_i^k$$

   where $|\mathcal{G}_i^k| = |\mathcal{G}_i|/5 = 5$.

2. **for** $k = 1$ to 5 `// round robin over the testing songs`
   - form the set of test songs

$$\mathcal{U} = \bigcup_{i=1}^{6} \{\mathcal{G}_i^k, \}$$

     and the set of training songs

$$\mathcal{L} = \bigcup_{i=1}^{6} \{\mathcal{G}_i^l,\ l = 1, \cdots, 5, l \neq k\}$$

   - test the performance of your algorithm on the 30 songs in $\mathcal{U}$ using the 120 training songs $\mathcal{L}$
   - record the confusion matrix $R^{(k,n)}$

   end for

end repeat

- **for** each entry (i, j) of the confusion matrix:
  - compute the average $\overline{R}(i, j)$
  - compute the standard deviation $\sigma_R(i, j)$

end for

The final average confusion matrix, and the associated standard deviation should be included in the report.