Lab 2
Lab report due by 2:00 PM on Monday, February 15, 2016

This is a MATLAB only lab, and therefore each student needs to turn in her/his own lab report and own programs.

---

1 Introduction

This is the second lab on digital signal processing methods for content-based music information retrieval. In this lab we will explore higher level musical features such as rhythm, and tonality. In the next lab you will be asked to classify tracks according to these features.

2 Rhythm

Musical rhythm can be loosely defined as the presence of repetitive patterns in the temporal structure of music. We will implement several techniques to detect the presence of rhythmic structures in an audio track.

Given an audio segment of duration T =**24 seconds**, the goal is to compute a score that measures how often spectral patterns are repeated. Formally, we will compute a vector B, such that `B(lag)` quantifies the presence of similar spectral patterns for frame that are `lag` frames apart. The lag associated with the largest entry in the array B is a good candidate for the period in the rhythmic structure.

We explain below two approaches to compute the array B.

2.1 Spectral decomposition

We first compute the mfcc coefficients, as explained in lab 1. We modify the `mfcc` function to add a nonlinearity. Indeed, the perception of loudness by the human auditory system is nonlinear: loudness does not increase linearly with sound intensity (measured, for instance in Watts/m^2). A good approximation is obtained by taking the logarithm of the energy. The `mfcc` measured in decibels (dB) is thus defined by,

$$10 \log_{10}(\texttt{mfcc}). \tag{1}$$

Your function should return an array `mfcc(k,n)` of size

$$\texttt{nbanks} \times N_f, \tag{2}$$

where $N_f = f_s T/K$ is the number of frames, K=256 is the number of frequencies, and `nbanks = 40` is the number of mels. The entire analysis is performed at the level of the frames.

2.2 Spectrum histogram

We propose to construct a visual representation of a musical track by counting how often a given note (corresponding to one index of the `mfcc` array) was played at a given amplitude, measured in dB.

Assignment

1. Modify your function that computes the `mfcc` to return the `mfcc` in decibels.
2. Construct a two-dimensional *spectrum histogram* (a matrix of counts), with 40 columns – one for each mfcc index, and 50 rows – one for each amplitude level, measured in dB, from -20 to 60 dB. You will normalize the histogram, such that the sum along each column (for each "note") is one.
3. Display, using the MATLAB function `imagesc`, the spectrum histogram for the 12 audio tracks supplied for the first lab.

2.3 Similarity matrix

The first stage involves computing a similarity matrix defined as follows

$$S(i,j) = \frac{\langle \texttt{mfcc}(:,i),\ \texttt{mfcc}(:,j)\rangle}{\|\texttt{mfcc}(:,i)\|\,\|\texttt{mfcc}(:,j)\|} = \sum_{k=1}^{\texttt{nbanks}} \frac{\texttt{mfcc}(k,i)\,\texttt{mfcc}(k,j)}{\|\texttt{mfcc}(:,i)\|\,\|\texttt{mfcc}(:,j)\|}. \tag{3}$$

$S(i,j)$ measures the cosine of the angle between the spectral signatures of frames i and j. We note that `mfcc` is always positive, and therefore $0 \leqslant S(i,j) \leqslant 1$.

The similarity $S(i,j)$ is large, $S(i,j) \approx 1$, if the notes being played in frame i are similar to the notes being played in frame j. We note that this similarity is independent of the loudness of the music.

Assignment

4. Implement the computation of the similarity matrix, and display the similarity matrices (color-coded) for the 12 audio tracks supplied for the first lab. See Fig. 1 for an example of a similarity matrix.

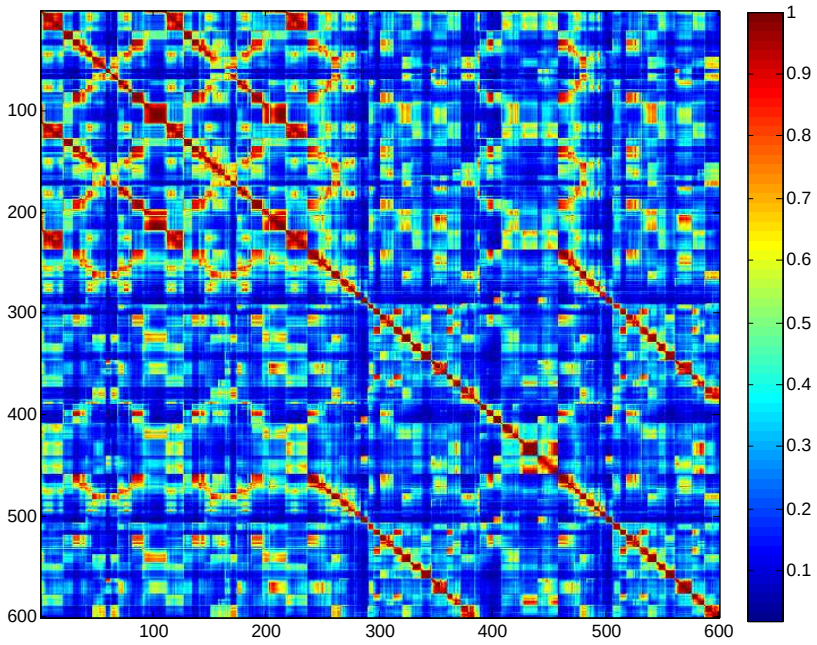

Figure 1: Similarity matrix associated with the sample `sample1.wav`. The matrix is symmetric: $S(i,j) = S(j,i)$

2.4 A first estimate of the rhythm

We note that the entries $S(n, n+l)$ in the similarity matrix (3) are always at a distance (time lag) of l frames. In order to detect the presence of patterns that are repeated every l frames we compute,

$$B(l) = \frac{1}{N_f - l} \sum_{n=1}^{N_f - l} S(n, n+l), \quad l = 0, \ldots, N_f - 1 \tag{4}$$

$B(l)$ is the sum of all the entries on the l^{th} upper diagonal. Indeed, $B(0)$ is the sum of all the entries on the main diagonal. Similarly, $B(1)$ is the sum of the entries on the first upper diagonal, and so on and so forth. We note that

$$0 \leqslant B(l) \leqslant 1. \tag{5}$$

The index l associated with the largest value of $B(l)$ corresponds to the presence of a rhythmic period of $l \times K/f_s$ seconds.

Assignment

5. Implement the computation of the rhythm index $B(l)$ defined in (4). Plot the vector B as a function of the lag $l = 0, N_f - 1$ for the 12 tracks. Comment on the presence, or absence, of a strong rhythmic pattern.
   Hint: to debug your function, you can use track-396 which should display strong rhythmic structure.

2.5 A better estimate of the rhythm

Rather than compounding the similarity between frames that are at a fixed lag apart in time, we can try to detect more interesting patterns in the similarity matrix. In general, if two frames i and j are similar, we would like to know if this similarity is repeated later in the segment, at time $j + l$. A lag l will be a good candidate for a rhythmic period, if there are many i and j such that if $S(i, j)$ is large then $S(i, j + l)$ is also large. This leads to the computation of the autocorrelation,

$$AR(l) = \frac{1}{N_f(N_f - l)} \sum_{i=1}^{N_f} \sum_{j=1}^{N_f - l} S(i, j)S(i, j + l), \quad l = 0, \ldots, N_f - 1. \tag{6}$$

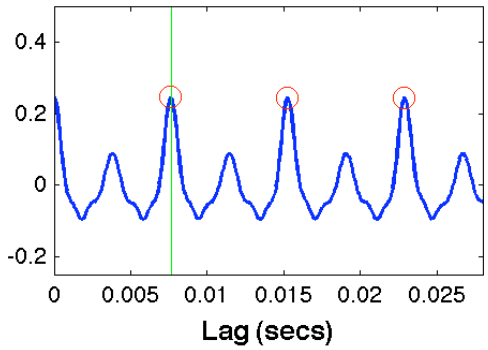

Figure 2: Rhythm index $AR(l)$as a function of the lag l

Assignment

9. Implement the computation of the rhythm index $AR(l)$ defined in (6). Plot the vector AR as a function of the lag $l = 0, N_f - 1$ for the 12 tracks. Comment on the presence, or absence, of a strong rhythmic pattern.
   See Fig. 2 for an example of a plot of the rhythm index.

2.6 Rhythmic variations over time

Here we are interested in studying the dynamic changes in the rhythm. For this purpose, we consider short time windows formed by 20 frames (approximately 1 second) over which we compute a vector AR of size 20,

$$AR(l, m) = \frac{1}{20(20 - l)} \sum_{i=1}^{20} \sum_{j=1}^{20-l} S(i + m * 20, j + m * 20)S(i + m * 20, j + m * 20 + l), \tag{7}$$

$$\text{for } l = 0, \ldots, 19, \quad \text{and } m = 0, \ldots, N_f/20 - 1. \tag{8}$$

Assignment

10. Implement the computation of the rhythm index $AR(l, m)$ defined in (8). Plot the image AR in false color as a function of the lag $l = 0, \ldots, 19$ (y-axis) and the time window index m (x-axis) for the 12 tracks. Comment on the variation of rhythm patterns for the different tracks.

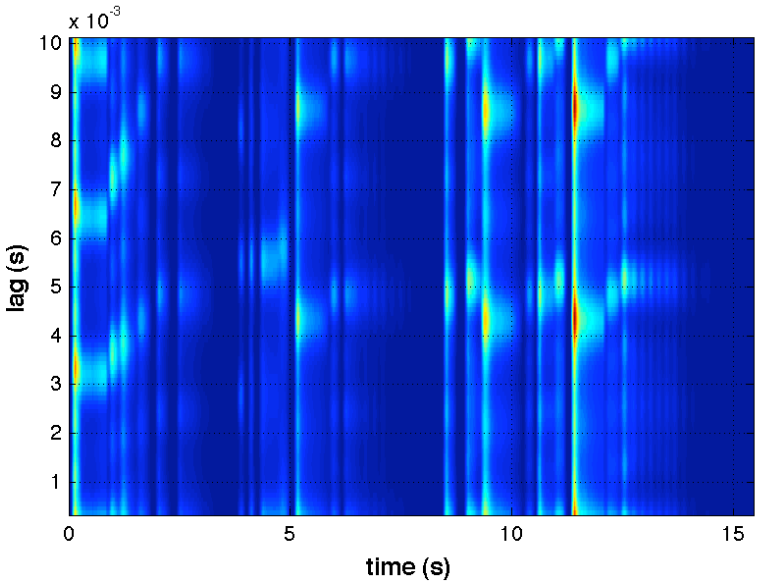

Figure 3: Rhythm index $AR(l, m)$as a function of the lag l (y-axis) and frame index m (x-axis).

3 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 the previous lab: the bark and the mel scales.

In this lab, 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 $f_0 = 27.5$Hz (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

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

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.

3.2.1 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 the first lab. We obtain $K = N/2 + 1$ (where N is even) Fourier coefficients, given by

$$\begin{aligned} Y &= \text{FFT}(w. * x_n); \\ K &= N/2 + 1; \\ X_n &= Y(1 : K); \end{aligned} \tag{13}$$

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 $f_k, k = 1, \ldots, n_{peaks}$ these peak frequencies.

3.2.2 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

$$sm = \text{round}(12 \log_2(f_k/f_0)). \tag{14}$$

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

$$c = sm \mod (12). \tag{15}$$

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

3.2.3 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,

$$PCP(c) = \sum_k w(k, c)|X_n(k)|^2, \quad \text{where } k \text{ is such that } c = \text{round}(12 \log_2(f_k/f_0)) \bmod 12, \tag{16}$$

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

$$w(k, c) = \begin{cases} \cos^2(\pi\, r/2) & \text{if } -1 < r = 12 \log_2(f_k/f_0) - sm < 1, \\ & \text{where } c = sm \mod (12), \text{ and } sm = \text{round}(12 \log_2(f_k/f_0)), \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

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.

3.2.4 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

$$NPCP(c) = \frac{PCP(c)}{\sum_{q=0}^{11} PCP(q)} \tag{18}$$

Assignment

11. Implement the computation of the Normalized Pitch Class Profile, defined by (18). You will compute a vector of 12 entries for each frame n.

12. Evaluate and plot the NPCP for the 12 audio tracks supplied for the first lab.
    See Fig. 4 for an example.

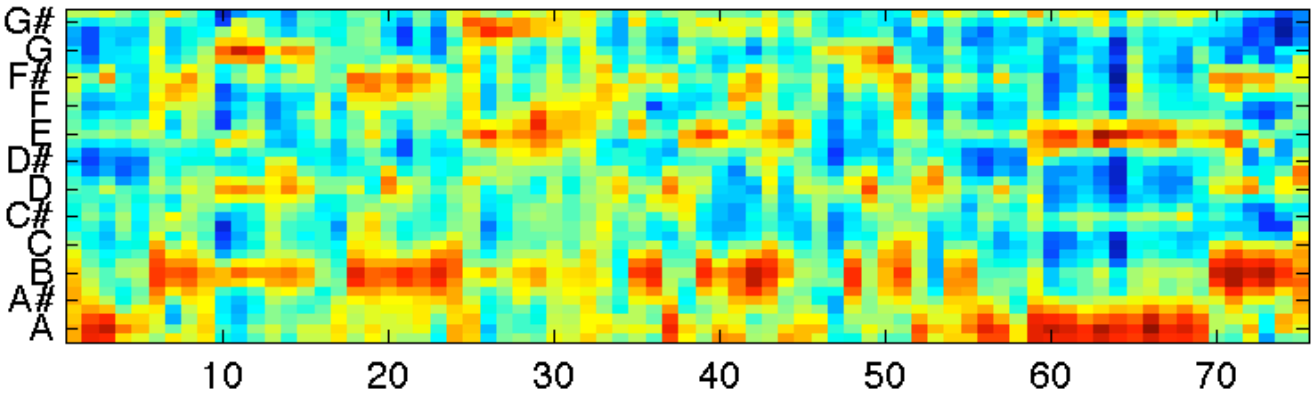

Figure 4: Chroma as a function of time (seconds).