Homework 1

Key finding algorithms: global key and local key detection of audio and symbolic music

Li Su
Institute of Information Science, Academia Sinica, Taiwan
lisu@iis.sinica.edu.tw

(You don't need any solid knowledge of music theory before doing this homework. So, don't worry if you are not familiar of the theory of musical tonality, mode, or key – taking this assignment will always help you learn it!)

Tonality, or so-called music *key*, is one of the most important attributes in music. In brief, key refers to two aspects of a musical scale: tonic and mode. The tonic is the first note of a diatonic scale. The mode is usually known as 'major' key, 'minor' key or so. Tonality is identified by the tonic note and the tonic chord. A very quick way to identify the key of a music piece is that, the tonic note is recognized as the *first* or the *last* note of a music piece. Moreover, if the chord corresponding to the tonic (i.e. the tonic chord) is a major chord, the music piece is then in major key. On the other hand, if the tonic chord is a minor chord, the music piece is then in minor key. However, this over-simplified way in finding key usually fails in most of the real-world musical data. Instead, as a high-level semantic object, key should be retrieved in a context-dependent manner. According to the musical context, a music piece may have a global key as its main key, and local keys which may change several times over a music piece. The detection of global and local keys is still not yet a solved problem in MIR.

In this assignment, we will design some global and local key detection algorithms for global and local music key detection for both audio and symbolic data, with full or limited contextual information. Therefore, you will learn how to extract audio features, how to deal with MIDI data, and also the music theory of tonality and its computational aspects.

The concept of a musical key

Let's start from the notion of the major and minor scales. Denote T as a tone and S a semitone, a major scale is a note sequence represented as T-T-S-T-T-T-S while a minor scale is T-S-T-T-S-T-T. The *functions* of these seven notes are tonic, supertonic, median, subdominant, dominant, submediant, and leading tone, respectively (see Figure 1). The major and minor scales are the two most commonly seen diatonic scale. If the tonic of a major scale is `C`, we then call it a `C` major scale. If the tonic of a minor scale is `C`, we then call it a `C` minor scale (see Figures 2 and 3).

A major scale and a minor scale that have the same tonic are called *parallel keys*. For example, the parallel minor of a C major key is a `C` minor key. A major scale and a minor scale that have the same key signatures are called *relative keys*. have For example, the relative minor key of the `C` major key is the `A` minor key, the relative minor of `E` major is `C#` minor, etc.

(PS: Do not confuse the major/minor key with the major/minor chord. A chord is the co-occurrence of (usually 3) notes, like the major triad and the minor triad, while a key represents the structural information in a diatonic scale.)

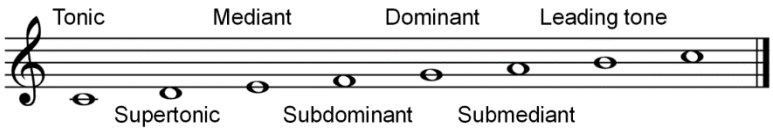

Figure 1: the diatonic scale. (Figure from:)

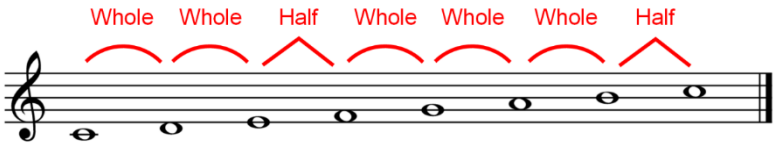

Figure 2: the C major scale.

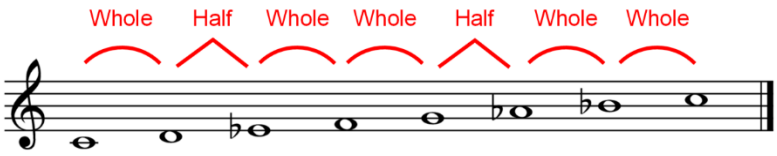

Figure 3: the C minor scale.

Prerequisite:

Install `librosa`, a Python library for music and audio signal processing: http://librosa.github.io/librosa/
Install `pretty-midi`, a Python library for MIDI signal processing: http://craffel.github.io/pretty-midi/#
Install `mir_eval`, a Python library for MIR evaluation: https://craffel.github.io/mir_eval/#

Download the GTZAN dataset and Alexander Lerch's annotation of key on the **GTZAN** dataset:
[Dataset] https://drive.google.com/open?id=1Xy1AIWa4FifDF6voKVutvmghGOGEsFdZ
[Annotation] https://github.com/alexanderlerch/gtzan_key
We will use the data from the following 9 genres for experiment: blues, country, disco, hip-hop, jazz, metal, pop, reggae, and rock.

Download the **GiantSteps** dataset and annotation:
[Dataset and annotation] https://drive.google.com/drive/u/3/folders/1D-PKkNWkWIQYcUDQokdzAFU0EL-oa3lc

Download the audio part of Beethoven's Piano Sonata Functional Harmony (**BPS-FH**) dataset:
[Dataset and annotation] https://drive.google.com/drive/u/3/folders/1I1cZ4RpGW6WB0goajxqJpX0u6h-HpmZR

Download the **A-MAPS** dataset and Adrien Ycart and Emmanouil Benetos' annotation on local key:
[Dataset and annotation] http://c4dm.eecs.qmul.ac.uk/ycart/a-maps.html#contents

Task 1: Binary template matching for global (or clip-level) key detection for audio data

In this method, we assume that the tonic pitch is the one which *appears the most often*. Therefore, a simple way of finding the tonic pitch is to 1) summing up all the chroma features of the whole music piece into one chroma vector (this process is usually referred to as sum pooling), 2) finding the maximal value in the chroma vector, and (3) considering the note name corresponding to the maximal value as the tonic pitch. Given a chromagram $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]$, $\mathbf{z}_i \in \mathbb{R}^{12}$, where N is the number of frames, the accumulated chroma vector of $\mathbf{Z}$ over the time is

$$\mathbf{x} = \sum_{i=1}^{N} \mathbf{z}_i$$

Knowing the tonic, the next step is to find the mode. Based on the idea of template matching, this can be done by finding the correlation coefficient between the summed chroma vectors and the mode templates. For example, if we have found that the tonic note is `C`, then we generate two mode templates, one for the `C` major mode and the other for the `C` minor mode:

$$\mathbf{y}_{\text{C Major key}} = [1\,0\,1\,0\,1\,1\,0\,1\,0\,1\,0\,1]$$
$$\mathbf{y}_{\text{c minor key}} = [1\,0\,1\,1\,0\,1\,0\,1\,1\,0\,1\,0]$$

The first index of $\mathbf{y}$ indicates `C` note, the second index `C#` note, …, and the last index `B` note. The correlation coefficient between the accumulated chroma and the template is:

$$R(\mathbf{x}, \mathbf{y}) = \frac{\sum_{k=1}^{12}(\mathbf{x}_k - \bar{\mathbf{x}})(\mathbf{y}_k - \bar{\mathbf{y}})}{\sqrt{\sum_{k=1}^{12}(\mathbf{x}_k - \bar{\mathbf{x}})^2 \sum_{k=1}^{12}(\mathbf{y}_k - \bar{\mathbf{y}})^2}}$$

where x is the summed chroma vector and y is the template for a key. (Hint: you may use some existed package such as `scipy.stats.pearsonr` in the `scipy` library) There are 24 possible keys, and according to Alexander Lerch's annotation, they are indexed as (upper case means major key and lower case means minor key):

A	A#	B	C	C#	D	D#	E	F	F#	G	G#
0	1	2	3	4	5	6	7	8	9	10	11
a	a#	b	c	c#	d	d#	e	f	f#	g	g#
12	13	14	15	16	17	18	19	20	21	22	23

If the tonic number is given as $0 \le j \le 11$, we only have to compare the correlation coefficients between $R(\mathbf{x}, \mathbf{y}^{(j)})$ (major key) and $R(\mathbf{x}, \mathbf{y}^{(j+12)})$ (minor key). If $R(\mathbf{x}, \mathbf{y}^{(j)}) > R(\mathbf{x}, \mathbf{y}^{(j+12)})$, we say the music piece is in the j major key, otherwise it is in j minor key. A music piece only has one key. The accuracy of key finding can therefore be define as:

$$\text{ACC} = \frac{\text{\# of correct detection}}{\text{\# of all music pieces}}$$

You may also consider some other existed package to compute the accuracy.

Q1 (15%): Compute the chroma features of each music clip using `librosa.feature.chroma_stft`. Use the binary template matching method mentioned above to find the key of all pieces of the GiantSteps dataset, and the 9 genres in the GTZAN dataset (since there is no annotation in the classical genre, you don't need to run that genre). Compare your estimation with the ground truth annotation of key in the dataset. What's the overall accuracy (ACC) of the two datasets? What are the ACC values for each genre? (Note: since GiantSteps has too many genres so you just need to report the overall accuracy.) Which genres have lower accuracy and can you guess why (from musical point of view)?

Note that some of the music pieces have unknown key labels, so please don't count these pieces while calculating the accuracy.

Q2 (15%): Set the *factor of logarithmic compression* γ. This factor is used in the nonlinear transform $\log(1+\gamma|x|)$ mentioned in our course slides. Adjust the factor of logarithmic compression to different values, say, $\gamma = 1$, 10, 100, and 1000. Repeat the experiment in Q1 and discuss how this factor is related to the result.

Q3 (20%): You might have found that some of the error detection results behave similarly. For example, `C` major key is easily to be detected as `G` major key (a perfect-fifth error), `A` minor key (a relative-major/minor error), or `C` minor key (a parallel-major/minor key), because these erroneous keys are intrinsically "closer" to `C` major keys than others. Therefore, in MIREX key detection competition, these closely related keys are considered in the scoring of key detection:

Relation to correct key	Points
Same	1.0
Perfect fifth	0.5
Relative major/minor	0.3
Parallel major/minor	0.2
Other	0.0

Therefore, the new accuracy is defined by:

$$\text{ACC} = \frac{\#\text{ Same} + 0.5(\#\text{ Fifth}) + 0.3(\#\text{ Relative}) + 0.2\ (\text{Parallel})}{\#\text{ of all music pieces}}$$

Use this new accuracy to evaluate the experiment in **Q1** and discuss the result. Note that you can directly use the evaluation function `mir_eval.key` in the `mir_eval` library.

Task 2: Krumhansl-Schmuckler key-finding algorithm

A more advanced set of templates for key detection is the Krumhansl-Schmuckler (K-S) profile. Instead of using a binary (0 or 1) templates as we did before, we assign values to the template according to human perceptual experiments. The template values are shown in the following Table (see the columns labeled by K-S). The experiment is done by playing a set of context tones or chords, then playing a probe tone, and asking a listener to rate how well the probe tone fit with the context.

Therefore, in Method 2, we consider using the correlation coefficient between the input chroma features and the K-S profile for key detection. Notice that the major and minor templates are rendered by different values. In this task we don't need to probe the tonic first, but just need to find the maximal correlation coefficient among the major profile, minor profile, and the 12 circular shifts of them, respectively. A web resource http://rnhart.net/articles/key-finding/ nicely demonstrates this idea.

Q4 (25%): Use the Krumhansl-Schmuckler's method to do the same task in **Q1**, **Q2**, and **Q3** and discuss the experiment result. Discuss the experiment results: which feature is better? Is there any limitation of these method? Is there any limitation of using GTZAN dataset for key finding? (Bonus: you may also compare your results to the other two features provided by `librosa`: they are `chroma_cqt` and `chroma_cens`)

Major key			Minor key		
Name	Binary	K-S	Name	Binary	K-S
Tonic	1	6.35	Tonic	1	6.33
	0	2.23		0	2.68
Supertonic	1	3.48	Supertonic	1	3.52
	0	2.33	Mediant	1	5.38
Mediant	1	4.38		0	2.60
Subdominant	1	4.09	Subdominant	1	3.53
	0	2.52		0	2.54
Dominant	1	5.19	Dominant	1	4.75
	0	2.39	Submediant	1	3.98
Submediant	1	3.66		0	2.69
	0	2.29	Leading tone	1	3.34
Leading tone	1	2.88		0	3.17

Task 3: Local key detection

In the previous task, we assume that one music piece has only one key. This is however not the case for Western classical music, where *key modulation* is heavily used. That means, the key of a music piece is likely to change over time. In the following task, we will design a key detector that can find the local key, and we will evaluate the algorithm on both audio and symbolic datasets.

We consider two datasets: the BPS-FH dataset containing synthesized piano of Beethoven's piano sonatas, and the A-MAPS dataset containing MIDI files. For the processing of the A-MAPS dataset, you may use the `pretty_midi` library and do the following things:

1. Use `pretty_midi.Instrument.get_chroma` to get the chroma vector.
2. Use `.key_signature_changes` to get the key and time when the key changed.

Q5 (25%): Based on Task 1 and Task 2, design a local key detector that outputs the key of the music every second. That means, there is a pitch detection output for every time step, and in this task, we set the time step be 1 second. Note that the speed of every sonata is modified to crotchet = 60. The output should therefore be like in the following format, where the first column is time (in seconds), and the second column is the detected key name (or number):

1	F
2	F
3	F
4	F
…	…
31	F
32	A-
33	A-
…	…

Figure: an example of local key data of Beethoven's Piano Sonata No. 1 in the BPS-FH dataset.

The local key detection accuracy is defined as

$$\text{ACC} = \frac{\#\text{ of correct detection}}{\#\text{ of time instances (detections) in all music pieces}}$$

$$\text{ACC} = \frac{\#\text{ Same} + 0.5(\#\text{ Fifth}) + 0.3(\#\text{ Relative}) + 0.2\ (\text{Parallel})}{\#\text{ of all time instances (detections) in all music pieces}}$$

Note that these accuracies count the number of time instances rather than the number of pieces.

If you use machine-learning-based method, please follow the train-validation-test partition:

BPS-FH	Train	1, 3, 5, 11, 16, 19, 20, 22, 25, 26, 32
	Validation	6, 13, 14, 21, 23, 31
	Test	8, 12, 18, 24, 27, 28
A-MAPS	Train	a~l (more information)
	Validation	m~p
	Test	s~z

You only need to report the accuracy of the test fold.

Hint: to get the local tonality feature, you may consider the accumulated chroma of a segment (maybe 30 seconds or so), not of the whole music piece. For example, the feature representing the local key at the 60th second can be obtained by summing up the chorma vectors from the 45th to the 75th second. In this case, what is the optimal segment size?

Q6 (bonus): if possible, please design an algorithm that (hopefully can) outperforms the two algorithms introduced here in at least two of the five genres considered in this assignment. You may use more advanced method (e.g., deep learning) and novel data representations that you may want to create.

The grading policy (e.g. about delay in HW submission) is the same as HW1. Please submit your .zip file containing the report (PDF) and your codes, with title "HW1 [your ID]" to the course website.

The deadline of Assignment 1 is April 20, and we will discuss it on May 5.