Chord2symbol Algorithm

Tongyu Lu, 26/Jun/2022

Main Framework

Algorithm: chord2symbol

Input: list of chord notes $\mathbf{c}$, list all controller "list_all"

Output: list of (root note, chord quality, bass note)

Initialize results with original-position detection

1. Parse chord notes into bass and chroma24: $b, \tilde{\mathbf{c}} = \text{to\_bass\_chroma24}(\mathbf{c})$
2. Get all possible chord qualities and importance weights from chroma24: $\mathbf{q}, \mathbf{w} = \text{chroma24\_to\_chord\_quality}(\tilde{\mathbf{c}})$
3. Initiate bass vector and root vector $\mathbf{b} = [b, \dots, b], \mathbf{r} = [\text{None}, \dots, \text{None}],\ where\ |\mathbf{b}| = |\mathbf{r}| = |\mathbf{w}| = |\mathbf{q}|$
4. If $(b, \tilde{\mathbf{c}})$ is a "bass-modified" chord (e.g., Bdim/C):
5. $\quad b', q', r' = \text{get\_bass\_modifed\_chord}(b, \tilde{\mathbf{c}})$
6. $\quad \mathbf{b} = [\mathbf{b}, b'], \mathbf{q} = [\mathbf{q}, q'],\ \mathbf{r} = [\mathbf{r}, r'], \mathbf{w} = [\mathbf{w}, w_{high}]$
7. Initiate result list $out = zip(\mathbf{b}, \mathbf{q}, \mathbf{r})$
8. Log initial bass note $b_0 = b$

Identify chord quality for each chord inversion

9. For $i = 0, \dots, |\tilde{\mathbf{c}}| - 1$:
10. $\quad b, \tilde{\mathbf{c}} = \text{invert}(b, \tilde{\mathbf{c}})$
11. $\quad \mathbf{q}', \mathbf{w}' = \text{chroma24\_to\_chord\_quality}(\tilde{\mathbf{c}})$
12. $\quad$ For q', w' in $zip(\mathbf{q}', \mathbf{w}')$:
13. $\quad\quad out = [out, (b, q', b_0)]$
14. $\quad\quad \mathbf{w} = [\mathbf{w}, w']$
15. If "list_all":
16. $\quad$ Return out
17. Else:
18. $\quad$ Return $out\left[\underset{i}{\text{argmax}}\, \mathbf{w}[i]\right]$

Chroma24 and Chroma12 Feature

- "chroma24" is a feature represented by a list of varying length, where components are in ascending order and are chosen in integer set $\{0,1, \dots ,23\}$.
- Chroma24 is a profile for a chord. It not only captures the note choices of a chord within an octave, but also captures the information of inversions.
- For example:
  - given a chord $[C4, E4, G4, C5, G5, E6]$, its chroma: $[4, 7]$
  - given a chord $[C4, D4, E4, G4, C5, G5, E6]$, its chroma: $[2, 4, 7]$
  - given a chord $[C4, E4, G4, C5, D5, G5, E6]$, its chroma: $[4, 7, 14]$
  - given a chord $[C4, E4, G4, B5, D5, E5, A5]$, its chroma: $[4, 7, 11, 14, 21]$
- As you can see, apart from third intervals, fifth intervals, seventh intervals and octave intervals, all other intervals are

reduced into either the first octave or the second octave in the chroma feature.

Here, we introduce the algorithm to convert pitch list into chroma24.

Algorithm: to_bass_chroma24

Input: list of pitches $c = [p_1, \dots, p_{|c|}]$

Output: the bass note of c and the chroma24 feature of c

1. If $|c| = 0$: return $-1, []$
2. If $|c| = 1$: return $p_1, []$
3. Sort c in ascending order
4. Initialize chroma24: $C = []$
5. For p in $[p_2, \dots, p_{|c|}]$:
6. $\quad \Delta p = p - p_1$
7. $\quad$ If $0 < \Delta p \leq 23$:
8. $\quad\quad \mathrm{chr} = \Delta p$
9. $\quad$ Elif $\Delta p \geq 24$:
10. $\quad\quad \mathrm{chr} = 12 + (\Delta p \bmod 12)$
11. $\quad$ Else:
12. $\quad\quad$ continue
13. $\quad$ If chr not in $C.\mathrm{append}(\Delta p)$:
14. $\quad\quad C.\mathrm{append}(\mathrm{chr})$
15. Return $p_1,\ \mathrm{rectify\_chroma24}(C)$

Algorithm: rectify_chroma24

Input: raw chroma24 $C = [I_1, \dots, I_{|c|}]$

Output: the rectified chroma24 C'

1. If $|C| = 0$: return C
2. If $|c| = 1$: return $p_1, []$
3. Initialize new chroma $C' = []$
4. For I in C:
5. $\quad \mathrm{chr} = I \bmod 12$
6. $\quad$ If $I \bmod 12 = 0$:
7. $\quad\quad$ Continue
8. $\quad$ Elif $I \leq 12$ and $\mathrm{chr} \notin C'$:
9. $\quad\quad C'.\mathrm{append}(\mathrm{chr})$
10. $\quad$ Elif I is third/fifth/seventh interval (including octaves like 10th) and $\mathrm{chr} \notin C'$:
11. $\quad\quad C'.\mathrm{append}(\mathrm{chr})$
12. $\quad$ Elif $I > 12$ and $\mathrm{chr} \notin C'$:
13. $\quad\quad C'.\mathrm{append}(I \bmod 24)$
14. Sort C' in ascending order and remove duplicate elements
15. Return C'

Chroma12 is nothing but converting the second-octave intervals into the first octave, getting the traditional chroma feature.

It is easy to observe that chroma24 is shift-invariant. It captures the chord quality information, regardless of the root note. For example, C_Δ^7 chord has the same chroma24 feature as G_Δ^7.

Chroma24 to Chord-quality Rules

The central part of chord2symbol algorithm is converting chroma24 into chord quality. We start from defining chord symbols, and the we design a mapping from chroma24 to the defined chord symbols.

Our system support the following chord qualities (examples in C):

1. All intervals
2. Major chords: $C, C^{2}, C^{4}, C^{\mathrm{b5}}, C^{\#5}, C^{\#4}, C^{6}, C^{7}_{\Delta}, C^{9}_{\Delta}, C^{67}_{\Delta}, C^{9}_{6}, C^{11}_{6}, C^{\#11}_{6}, C^{11}_{\Delta}, C^{7\#11}_{\Delta}, C^{13}_{\Delta}, C^{13\#11}_{\Delta}$
3. Minor chords: $C^{-}, C^{-2}, C^{-4}, C^{-6}, C^{-67}, C^{-9}, C^{-9}_{6}, C^{-11}_{6}, C^{-\#11}_{6}, C^{-11}, C^{-7\#11}, C^{-1}, C^{-1\ \#11}$
4. Major-minor chords: $C^{-\mathrm{M7}}, C^{-6\mathrm{M7}}, C^{-9\mathrm{M7}}_{6}, C^{-\mathrm{M7\#11}}_{6}, C^{-9\mathrm{M7}}, C^{-11\mathrm{M7}}, C^{-\mathrm{M7\#11}}, C^{-13\mathrm{M7}}, C^{-13\mathrm{M7\#11}}$
5. Dominant chords:

$$C^{7}, C^{9}, C^{11}, C^{13}, C^{7\mathrm{b9}}, C^{11\mathrm{b9}}, C^{13\mathrm{b9}},$$
$$C^{7\#5}, C^{9\#5}, C^{11\#5}, C^{13\#5}, C^{7\#5\mathrm{b9}}, C^{11\#5\mathrm{b9}}, C^{13\#5\mathrm{b9}}, C^{7\#5\#11}, C^{13\#5\#11},$$
$$C^{7\mathrm{b5}}, C^{9\mathrm{b5}}, C^{11\mathrm{b5}}, C^{13\mathrm{b5}}, C^{7\mathrm{b5b9}}, C^{11\mathrm{b5b9}}, C^{13\mathrm{b5b9}}$$

6. Altered chord: C_{alt}
7. Suspended chords: $C^{\mathrm{sus}}, C^{\mathrm{sus2}}, C^{\mathrm{sus7}}, C^{\mathrm{susM7}}, C^{\mathrm{sus13}}, C^{\mathrm{susM13}}$
8. Augmented chords: $C^{+}, C^{+\mathrm{M7}}, C^{+\mathrm{M7,M9}}, C^{+\mathrm{M7,M11}}, C^{+\mathrm{M7\#11}}, C^{+\mathrm{M7,M13}}, C^{+\mathrm{M7,M13\#11}}$
9. Diminished chords: $C^{o}, C^{o7}, C^{o\mathrm{M7}}, C^{o9}, C^{o\mathrm{M9}}$
10. Half-diminished chords: $C_{\phi}, C^{9}_{\phi}, C^{11}_{\phi}, C^{11\mathrm{b13}}_{\phi}$
11. Any extra notes added on the base of all chord symbols above

Now, we have the codomain of the chroma24-to-chord-quality mapping. Now, we design the mapping rules.

We proceed the mapping through a checklist, starting from the simplest chords to the most complex chords.

The checklist is as follows:

```
chord_quality_checklist = [
    is_intervals,
    is_major_basics,
    is_major_less_basics,
    is_minor_basics,
    is_minor_less_basics,
    is_dominant_basics,
    is_dominant_less_basics,
    is_sus_basics,
    is_aug_basics,
    is_dim_basics,
    is_dim_less_basics,
    is_alt,
    is_triad_extensions,
    is_major_seventh_sixth_extensions,
    is_halfdim_seventh_extensions,
    is_minor_seventh_sixth_extensions,
    is_dom_seventh_extensions,
    is_sus_seventh_extensions,
    is_aug_major_seventh_extensions,
     is_dim_seventh_extensions,
]
```

The rules are mutual exclusive. The rules are also carefully designed such that they accomplish a surjective from chroma24 to the defined chord symbols. For each basic rule, only chroma12 is considered because they range within an octave. For altered chord and extension chords, chroma24 is considered for extension determination (e.g., 6^{th} vs. 13^{th}). For inverted

chords, only basic chord inversions are considered, this is because extension chord inversions are almost surely not considered.

Each rule takes input arguments: chroma24 and inverted (bool flag). Each rule bears a set of templates for specific chord types. If a certain rule detects a specific chord type based on the given chorma24, it outputs a tuple indicating the chord type and its corresponding importance; if not detected, it returns a False flag.

The detailed template definitions are complicated. Check the code for details.