EXPLORING AUTOMATIC CHORD RECOGNITION ALGORITHMS

Shan Anis
University of Rochester
sanis@ur.rochester.edu

ABSTRACT

Automatic Chord Recognition can be described as the task to divide an audio file containing music, into frames and then assigning each frame a chord label according to the analysis of the content. Chord recognition can be very advantageous for a number of other applications involving music information retrieval e.g. Automatic Music Transcription. Most Automatic Chord Recognition algorithms implement Chroma (or Pitch Class Profile) Feature Extraction, Pre-Filtering, Pattern Matching and Post-Filtering, however, there is limited understanding in the development and optimization of these processes and the variables involved in their computation. The objective of this paper is to analyze current developed Automatic Chord Recognition Systems, perform a systematic evaluation to analyze the different stages of the typical system, experiment with combinations of different methods and variables in order to come up with improvements. Our aim is to better understand each stage of the system and propose new directions to potential improvements. In this work, the system built uses Chroma features for the feature extraction and Hidden Markov Models (HMM) for the Pattern Matching in a supervised training environment.

1. INTRODUCTION

Chords are defined by the occurrence of 3 or more harmonically related musical notes played either simultaneously or in quick succession (arpeggio). Chords define the fundamental structure of the tonal system in western music, therefore the ability to detect them can be very valuable for a variety of applications in music information retrieval.

The motivation to develop automatic chord recognition algorithms comes from the fact that although the task is not very difficult for trained musicians, it is time consuming, repetitive and almost impossible to label the entire vast database of songs one by one. Furthermore, the results can assist in music information retrieval tasks such as: genre classification, cover song identification, music structure analysis, music transcription, etc.

Since there are a total of over 100 different existing chords, the task can become quite complex and therefore, typically only the 12 major and 12 minor triads are considered. Some systems utilize a 25th 'no-chord' element in their defined vocabulary to detect frames without any harmonic content.

Using predefined templates for chord recognition can be a very time consuming task as it required manual annotation of the chord labels. In this work, we utilize HMMs in a supervised training environment to eradicate the need for manual annotations. This approach can also be used to detect chords other than the 24 majors and minors.

Our work is inspired from Bello and Pickens [1] where they also used Hidden Markov Models with the Expectation Maximization for the task of chord recognition. They utilized concepts in music theory to define the transition matrix based on the key distance in a circle of fifths.

2. SYSTEM FRAMEWORK

A typical chord recognition system consists of four stages: feature extraction, pre-filtering, pattern matching and post-filtering. This is shown in Figure 1. Feature extraction transforms an audio file into meaningful representations. Pre-filtering performs smoothing on features which blend a single feature with the nearby context. Pattern matching can be carried out by comparing the similarity between a feature and a set of pre-defined chord patterns and selecting the pattern which is most similar to the feature. The patterns can be manually defined or learned and the similarity can be computed by various distance approximation methods depending on the implementation strategy of the system.

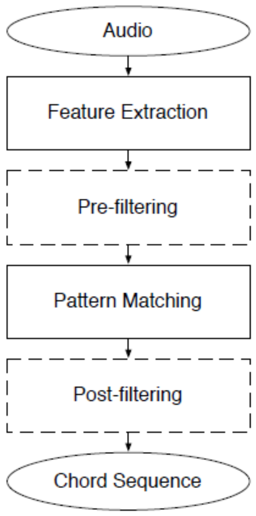


Figure 1 - Typical Chord Recognition System

2.1 Feature Extraction

An audio file contains a large set of music information such as: melody, harmony, beat, tempo, timbre of different instruments, dynamics of the sound, etc. For a specific music processing task, only some of this information is relevant and useful. Therefore we need to select the most important information related to the specific task. Usually we name this processing stage as feature extraction, being the procedure of transforming an audio file into a musically meaningful representation which keeps the most task-related musical properties while suppressing other unrelated information. At the same time, the form of the representation should be appropriate for the next processing stage.

For our tasks of identifying chords, we need to extract audio features that emphasize on the tonal structure and the musical properties of the audio file such as: chord progressions, melodies, component notes of a chord and its harmonics. A good feature should be able to capture the required music properties and also be invariant to some unrelated properties such as tempo and timbre. This is beneficial to us because it can allow accurate chord detection of different interpretations of a music piece played by different instruments (and hence, different timbres) at different speeds (different tempo).

Since a chord itself can be fully determined by its component notes, theoretically if the notes are known, the chord could be identified. Thus the capture of notes plays a crucial role in the feature extraction stage. Moreover, it is better to use pitch class instead of single notes to describe the form of the chord since human beings' perception of chords is irrelevant to the octave information of a note. Therefore our designed features should be able to project the information for notes within the same pitch class and distinguish the notes in different pitch classes.

2.1.1 Chroma Features

Chroma based audio features are a well-established tool in processing and analyzing music data and are particularly very suitable for the task of chord recognition.
As we know that human's perception of musical notes has a certain character: if a note is one or more octave higher than another note, then the two notes sound to have the same "tone color" but different "tone height". This phenomenon is referred to as octave equivalence in music theory. Assuming the musical notes are of equal-tempered scale, the chroma correspond to the set {C, C#, D. . . B} that consists of the twelve pitch classes.
For example, note A4 denotes the chroma as A and "tone height" as the 4th octave. We can reduce the MIDI notes from 88 pitches to 12 chroma classes by ignoring the octave information of the notes and classifying via chroma information.

Chroma features are very suitable for the task of chord recognition. This is because when analyzing a musical chord, we are much more interested in the chroma (or pitch class) which the note belongs to other than the absolute pitch of that note. For notes sharing the same chroma but in different octaves, we treat them as identical when considering a component note of the chord. Furthermore, different timbre of instruments will yield different yet distinctive energy distribution at harmonics, and since the energy of a chroma is merged from different pitches corresponding to this chroma together, the difference caused by timbre is well absorbed by chroma features, thus making them robust to the variations of timbre.

The chroma features are in the form of a 12-dimensional vector $x = (x(1), x(2), . . . , x(12))^T$, where x(1) corresponds to chroma C, x(2) to chroma C#, and so on. Chroma-based features represent the short-time energy of the signal in each of the 12 pitch classes. Often these chroma features are computed by suitably pooling spectral coefficients obtained from a short-time Fourier transform. Similarly, one can start with the pitch decomposition mentioned earlier. Then, by simply adding up the corresponding values that belong to the same chroma, one obtains a chroma representation or chromagram. Some systems apply L2 normalization to the resulting vectors.

Logarithmic Compression can be applied during the computation of Chroma Features to account for the logarithmic nature of sound intensity in the human auditory system. Furthermore, it is also advantageous to us since it adjusts the dynamic range of the original signal to enhance the clarity of weaker transients (similar to how a compressor works). This helps the identification of some weak chroma features that were difficult to detect due to their low intensities.

Most systems implementing the Chroma Key Feature Analysis utilize a frame rate which is significantly faster than the rate of chord changes in the audio input. This is required for accuracy. However, faster frame rates cause the system to be sensitive to noise and transients. To counter this problem, pre-filtering can be used.

For demonstration, we use an audio recording of a piano playing the chromatic scale and generate short time chroma frames (as columns). The figure below shows the spectrogram of the original wav file and its chromagram (on dB magnitude scale). A default value of 2048 was set for the 'fftlen'.

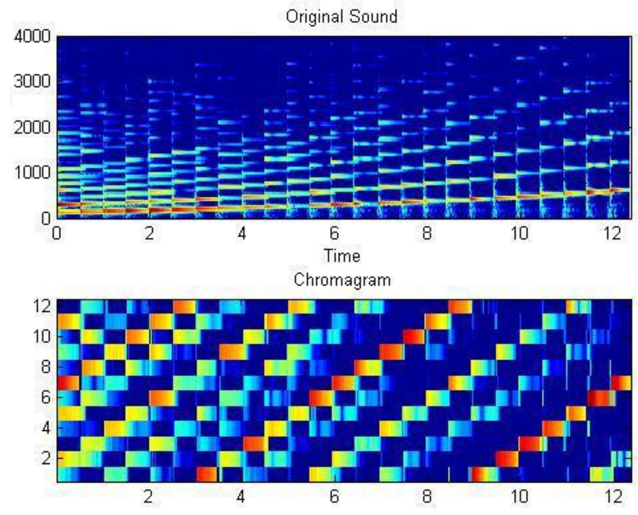


Figure 2 - Chromagram of the Chromatic Scale

In order to precisely identify chord boundaries, the frame rate of the chroma features must be faster than the rate of chord changes in the music. Using a longer window exaggerates the influence of transient noise and the disadvantage of using a short window is that the frames of the resulting chromagram are independent of the long term trend of the signal and respond to local changes, therefore becoming sensitive to transients and noise in the signal. To cope with this problem, pre-filtering is applied prior to pattern matching using a low pass filter. This technique improves pattern matching because it minimizes the effect of transients and noise in the signal by smoothing the features across neighboring frames.

2.2 Pattern Matching

After converting the audio file into musically meaningful audio features, we now pass these features into the chord recognition module which automatically classify the feature vectors with respect to given chord labels. The chord recognition module assigns a chard label to each feature vector.

In template based methods, first, pre-computed feature templates are defined that serve as chord templates. The templates can be defined in various ways (some of which are covered in this paper). Next, we need to find a distance measure between features and the pre-defined template and lastly, we assign the chord label by selecting the one which results in the minimum distance to the given feature.

2.2.1 Cyclic Shift

A common technique used in template sets is the Cyclic Shift. For each of the template sets, we only set two templates instead of setting all templates.
We set one for C and the other for Cm, and denote them as T_C and T_{Cm}. The templates for other major triads are computed by cyclically shifting T_C, and other minor triads are computed by cyclically shifting T_{Cm}. The reason of involving cyclically shift is to utilize the characteristics of the chords. Since the musical interval between the third and root, fifth and root are always fixed for the same type of chord, one can derive the same type of the chords by first changing the root note and then make sure the third and fifth note from the musical interval. Therefore, we can derive the templates for the same type of chords by cyclically shifting the position of the notes.

2.2.2 Binary Templates

A binary chord template is the simplest and one of the most popular chord models. This deterministic chord model is manually generated based on knowledge of the notes used in musical chords. In a binary chord template vector, each component corresponding to a chord-tone 2 is set to 1, and the other components are set to 0. While some systems use variations of the binary chord template that incorporate information about higher harmonics produced by each chord-tone, recent studies have shown that simple binary chord templates are sufficient to obtain a good level of accuracy. For example, C is composed by the note C, E and G; Cm is composed by C, D# and G. While designing the templates in this method, for every given chord, we only consider the component notes of chord and totally discard other non-component notes. Thus it is reasonable to involve a binary setting since a note is either a component or a non-component one.
Each template in the set is a 12-dimensional binary vector with three entries equal to one and other entries equal to zero. The three non-zero entries correspond to the three component notes of a chord.

For example, the binary template corresponding to C major (C, E, G) is given by:

$$T_C = (1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0)$$

And the binary template corresponding to C minor (C, D#, G) is given by:

$$T_{Cm} = (1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0)$$

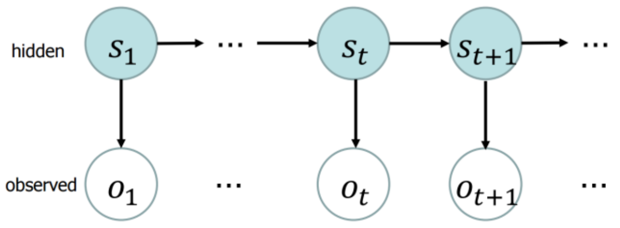
The advantage of the binary setting is its simplicity and efficiency. In contrast, the simplicity also leads to a limitation: it considers only the very ideal instance of a chord which works theoretically. However in practice, the intensity of the three component notes may not be exactly the same but very different. Also, it ignores too much information, for instance, it totally disregard the non-component notes, which may contribute to form the pattern of a chord.

2.2.3 Probabilistic Chord Models

Sophisticated chord models are created by defining probability distributions for each chord class. A popular choice is the multivariate Gaussian distribution. In some systems, the Gaussian chord models are defined manually as with binary templates. More commonly, the distribution parameters are estimated from labeled data.
More precise chord models in the form of Gaussian mixture models (GMM) are sometimes constructed instead of single Gaussian models. Such models use multiple Gaussian distributions to represent each chord. Different components represent more nuanced instantiations of each chord in the training data, producing a more precise fit. This comes at the cost of requiring more sophisticated training therefore requiring more computation power.

2.2.3.1 Hidden Markov Models

A Hidden Markov model is a statistical model in which the system being modeled is assumed to be a Markov process with hidden states. The output for each state corresponds to an output probability distribution. We can implement HMMs in the task of automatic chord recognition by considering chords as a hidden state in HMM and the features as observations. An HMM can be represented by its: initial probability, observation probability and transition probability. Initial probability is typically set to 1/24 (in a 24 chord recognition task) to give every chord a fair chance. The computed feature vectors serve as the observations and the transition probability is trained from the training data.



A single Gaussian in 12 dimensions is typically used to model the chroma vector distribution for each state. The Gaussian model is described by its mean vector and covariance metric.

The transition probability matrix describes the first order temporal relationships between the chord models. Each element of the matrix represents the probability of a chord switching to another.

2.3 Post-filtering

The post-filtering stage shown in Figure 1 is used to smooth the sequence of predicted chord labels over time, thereby minimizing the number of false chords that only last for a small number of frames. Such misdetections can be caused by short bursts of noise, which are very common in real music signals. In most systems, post-filtering is performed with a Viterbi decoder, while some systems based on chord templates use a median filter instead.

Once the initial probabilities, transition probabilities, observed probabilities and mean vector and covariance matrix for each state are learned, the Viterbi algorithm is applied to the model to find the optimal path. According to Sheh and Elis (2003) [2], *"The output of the Viterbi algorithm is the single state path labeling with the highest likelihood given the model parameters".*

The Viterbi algorithm is used to decode the hidden states of chords given the sequence observed feature vectors.

3. EXPERIMENTS

We evaluate the system variations using the well-known Beatles data set, 180 annotated songs from 12 Beatles albums (containing 13 discs). The ground truth chord annotations of the songs are hand labeled and provided by Christopher Harte. The evaluations are performed on 12 major, 12 minor and a no-chord detection task.

Kevin Murphy's Wonderful HMM Toolbox was used in HMM implementation, in particular, the *"gaussian_prob"* and *"viterbi_path"* functions were utilized.

Out of the 180 songs, 141 were used for the training and the remaining 39 were used for testing.

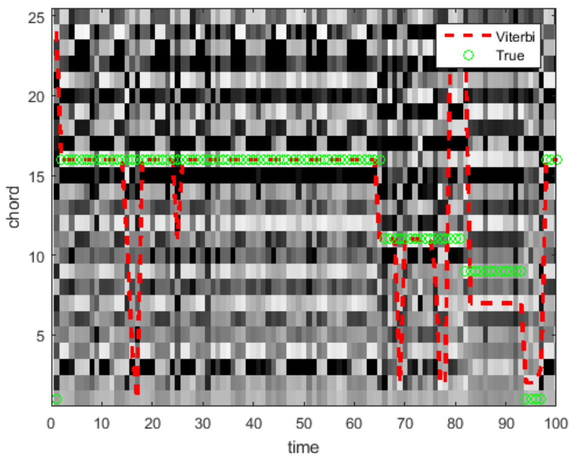


Figure 3 – Applying Viterbi Algorithm

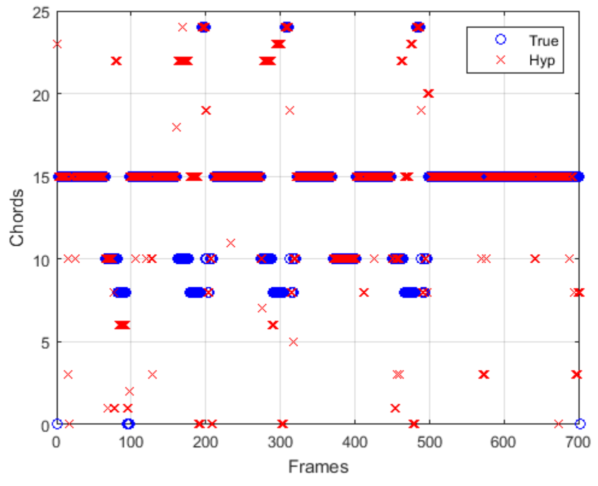


Figure 4 – Ground Truth Comparison

Figures 3 and 4 show the results obtained using the song "Come Together" as input.

Overall recognition accuracy for the entire testing set was 57.7%.

4. CONCLUSION

Experiments show that using Chroma Features with Hidden Markov Models in a supervised training environment can be used to solve the task of Automatic Chord Recognition. The greatest advantage of using HMMs is the fact that manual annotations are not required.

Although the recognition accuracy is not high enough at the moment for useful applications such as Automatic Transcription, we hope to improve these results in the future.

5. REFERENCES

[1] J. P. Bello and J. Pickens, "A robust mid-level representation for harmonic content in music signals," in Proceedings of the International Symposium on Music Information Retrieval, London, UK, 2005.

[2] Sheh and D. Ellis, "Chord segmentation and recognition using em-trained hidden markov models," in Proc. ISMIR, 2003.

[3] T. Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music," in Proc.ICMC, pp. 464–467, 1999.

[4] Sheh and D. Ellis, "Chord segmentation and recognition using em-trained hidden markov models," in Proc. ISMIR, pp. 185–191, 2003.

[5] Taemin Cho, Ron J. Weiss and Juan P. Bello, "Exploring Common Variations In State Of The Art Chord Recognition Systems"

[6] Christopher Harte, Mark Sandler, Samer A. Abdallah, and Emilia G´omez. "Symbolic representation of musical chords: A proposed syntax for text annotations."

[7] M. M¨uller, S. Ewert, and S. Kreuzer, "Making chroma features more robust to timbre changes,"