

Automated Identification of Chord Progression in Classical Music




Peiqian Li
Computer Science
Wittenberg University

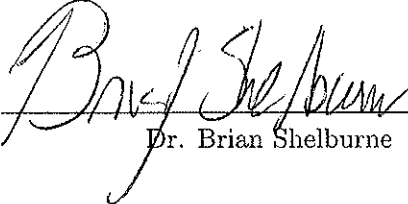
A thesis submitted for the degree of
B.S., Computer Science

May 2014


This thesis entitled:
Automated Identification of Chord Progression in Classical Music
written by Peiqian Li
has been approved for the Department of Computer Science



Dr. Steven Bogaerts



Dr. Brian Shelburne



Dr. Charles Grogan

4/18/2013
Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Abstract

This paper examines a supervised machine-learning algorithm for automatic chord sequence recognition from synthesized audio of Classical music phrases. A hidden Markov model-based system achieves 67.7% accuracy on an independent set of fifty selections. The system is trained on seventy-five phrases from compositions by Bach, Mozart, and Beethoven, that have been hand-labeled through symbolic analysis of the phrases in MIDI format.

Chapter 1

Introduction

Chord progression - the succession of chords over time - defines the harmonic structure in a musical composition. Once the chord progression of a particular musical piece is determined, this mid-level information about harmonic content can be used for higher-level analysis on the underlying musical signal. Harmonic analysis with detailed information on chord boundaries is very important for applications such as music search, segmentation, and similarity identification. For such reasons, many higher-level analyses of a musical composition usually start with labeling all chords of the piece.

Labeling chords by hand from music recordings can be tedious, error-prone, and time-consuming when this has to be done for a large number of compositions. Hence, automatic chord recognition systems are very useful for those who need harmonic information on a piece of music.

1.1 Chord

A chord in music is a harmonic set of at least three notes intended to be heard as if sounding simultaneously. The set of notes forming a chord are not necessarily played at the same time. Chords can be broken into groups of notes, or even individual notes played in sequence, one after another, as in *arpeggios*.

Chords consisting of three different notes are called **triads**. The three notes are the **root** (or **first**), the **third**, and the **fifth**, so called because the third note forms a third interval with the root note, and the fifth likewise. An interval can have one of the following qualities: perfect, major, minor, augmented, and diminished. Depending on the quality of the intervals formed by the third and fifth with the root note of a triad, the triad can be described as major, minor, augmented, or diminished. For example, a major triad is built by a major third and a perfect fifth above the root note; a diminished triad is formed by a minor third and a diminished fifth above the root note.

If another note is added to a triad to form a chord consisting of four notes, that extra note is usually the **seventh** above the root, thereby called the **seventh**. Such chords are named **seventh chords**. The quality of the seventh interval between the seventh and the root, together with the quality of the triad formed by the other three notes, determines the quality

of a seventh chord. For instance, a half-diminished seventh chord consists of a diminished triad and a minor seventh note above the root; a dominant seventh chord (or major-minor seventh chord) is made up of a major triad with a minor seventh note.

The following eight common chord qualities are considered in this paper:

Quality	Third	Fifth	Seventh	Example
major triad (M)	major	perfect	N/A	C-E-G
minor triad (m)	minor	perfect	N/A	C-E \flat -G
augmented triad (aug)	major	augmented	N/A	C-E-G \sharp
diminished triad (dim)	minor	diminished	N/A	B-D-F
diminished seventh (dim7)	minor	diminished	diminished	B-D-F-A \flat
half-diminished seventh (hdim7)	minor	diminished	minor	B-D-F-A
minor seventh (m7)	minor	perfect	minor	D-F-A-C
dominant seventh (dom7)	major	perfect	minor	G-B-D-F

Furthermore, a chord’s *inversion* refers to the relationship of its lowest note (bass) to the other notes in the chord. When the bass is not the root of a chord, the chord is described as *inverted*. We ignore chord inversions for our purposes, although chords in different inverted positions may play different harmonic roles in a composition.

1.2 Background and Related Work

Researchers in music information retrieval have shown that statistical machine learning methods work reasonably well for chord recognition. Some researchers focus on chord recognition from musical scores. For example, Perera and Kodithuwakku use an artificial neural network following the multiple adaptive linear neuron (MADALINE) network model to develop a system that works with music in Musical Instrument Digital Interface (MIDI) format [10]. Hidden Markov Models (HMMs) are very popular among studies of chord recognition from audio recordings. For example, Sheh and Ellis use an HMM trained by the expectation maximization algorithm to develop a statistical machine learning method for chord segmentation and recognition from audio recordings [4]. They use chord sequences as the sole input to their models and treat the chord labels as hidden values in an HMM. Their model includes 147 chord types, but their training data contain only 20 songs. The insufficient training may explain their 22% accuracy for chord recognition.

Bello and Pickens use a similar method, but they also incorporate some musical knowledge into their model initialization [1]. They build the key distance, according to the circle of fifths, into their state transition matrix. They consider 24 major and minor triads only, and achieve 75% accuracy. One of the many challenges commonly recognized in these studies is the lack of training data: music with labeled chord boundaries. In many later studies, researchers use the same training data set consisting of the twelve Beatles studio albums with chord transcriptions by Chris Harte [12]. HMM-based systems have been shown to perform reasonably well on these Beatles albums, with more than 70% accuracy of chord recognition achieved by a number of researchers, though many systems considered a limited number of chord qualities, such as only

major and minor triads.

Probably due in large part to the lack of availability of chord label databases for classical music, there has been little research on the application of statistical learning methods on chord recognition in classical music, i.e. music from the common practice period. To this end, we use methods similar to previous studies to explore the performance of an HMM-based chord recognition system on classical music compositions.

Chapter 2

Overview

Most HMM-based chord recognition systems proceed in a similar fashion. A given music recording is converted into a series of features that represent the audio spectrum. Then, a trained HMM is used to perform chord pattern matching, i.e. mapping the chroma features to chord labels that correspond to the various chords under consideration.

Figure 2.1 shows an overview of our system.

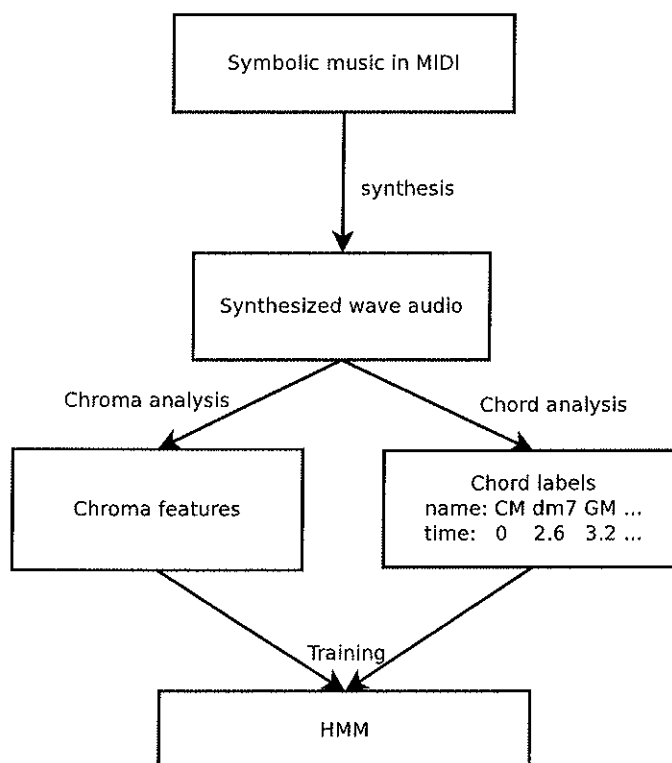


Figure 2.1: Overview of the chord recognition system

2.1 Music Selections and Synthesis

Since recordings of real performances inevitably include noise, we work with wave audio synthesized from MIDI symbolic data. The artificially synthesized audio clips can feature the enharmonic spectrum of musical instruments, making them comparable to recordings in the real world, but without extraneous noise.

Since many classical compositions are long, it is not feasible to manually label chords in a large number of pieces. In addition, most pieces contain repeated musical materials, and the same chord progression likely occurs multiple times. Therefore, we train and test our system with extracted phrases, instead of entire compositions. Repetition is avoided by strategically choosing parts of music with different harmonic content.

A total of 125 music selections are used to train and test the system. The training set contains 25 musical phrases each from J.S. Bach, W.A. Mozart, and L. van Beethoven. The other 50 selections from music by the same three composers are used to test the performance of the chord recognition system.

The MIDI files for our selections are obtained from Classical MIDI Connection [13] and Classical Piano Midi Page [14]. These symbolic files are synthesized by Timidity++ [15], a free software synthesizer, into WAVE format. Timidity++ comes with digital instrument data files, and uses a sample-based synthesis technique to generate enharmonically rich audio as in real recordings [9].

2.2 Chord Analysis

In order to train a supervised model like an HMM, it is necessary to annotate chord progression information in each selection. Chord labels include both chord names (root and quality, e.g. F major) and boundaries (start and end time relative to the beginning of the selection).

This is a laborious process, but two factors helped speed up chord-labelling. First, the Melisma Music Analyzer by Sleator and Temperley¹ was used to extract various kinds of information such as phrase structure, meter, harmony, and key. Although the automatically extracted information is not always accurate, it usually gives correct chord boundaries for a significant portion of each selection.

Second, some of the selections are used by Kostka and Payne in their text [11] as examples. For these selections, we only need to align the chords shown in the book in accordance to our synthesized recordings.

2.3 Chroma Analysis

Raw audio, synthesized or not, needs to be converted into a more useful and convenient representation before it can be processed and analyzed by a statistical model like an HMM. Chroma features are very powerful in presenting information necessary for chord analysis. All

¹<http://www.link.cs.cmu.edu/music-analysis/>

synthesized selections are converted into chroma features before they are presented to the HMM.

Chroma features are discussed in details in the following chapter.

Chapter 3

Chroma Features

Chroma features, also known as pitch class profiles, are well established in music audio processing applications [7]. The *chroma* correspond to the set of twelve pitches in the equal-tempered scale: $\{C, C\sharp(D\flat), D, D\sharp(E\flat), \dots, B\}$. Considering music audio as signals, we can characterize the short-time energy distribution of the underlying signals over the twelve (equal-tempered) chroma bands, usually by short-time Fourier transforms. Chroma features are often represented by a twelve-dimensional vector $f = (f_1, f_2, \dots, f_{12})^T$. A visual representation of chroma features over time is called a *chromagram*.

Note that the *height* of a pitch, telling us which octave the pitch belongs to, is not captured by the *chroma*. Chroma features only represent the relative intensity in each of the twelve semitones (how strong each pitch class is present in the signal), regardless of their heights, in a short time window. Since a chord label only depends on chroma, regardless of heights, chroma features are an ideal representation for harmonic analysis.

3.1 Chroma-Pitch Features

As the first step of chroma feature extraction, the given wave audio is divided into short segments (frames) of fixed length with overlap. We use a window length of 200 milliseconds and an overlap of 50%, leading to a frame rate of 10Hz (10 frames per second). A chroma feature vector is calculated for audio signals within each frame. Therefore, we have 10 feature vectors per second.

Within each frame window, the audio signal is decomposed into 88 frequency bands with center frequencies corresponding to the pitches A0 to C8 (MIDI pitches 21 to 108). For each band, we calculate the short-time mean-square power. The results tell us the local energy of each pitch band, and indicate how strong a certain note is present at a certain time. Values that belong to the same chroma are then added together; for example, the energy of A0, A1, A2, etc. all add up to the value for A. The resulting twelve-dimensional vector contains Chroma-Pitch (CP) features.

3.2 Chroma-Log-Pitch Features

Since sound intensity is perceived logarithmically rather than linearly, a logarithmic compression can be applied before adding up the energy levels of individual pitch bands. Feature vectors obtained this way are referred to as chroma-log-pitch (CLP) features. CLP features are used in our system.

3.3 Implementation

We performed CLP feature extraction on our entire data set using the Chroma Toolbox implemented in MATLAB by Müller and Ewert [8]. Figure 3.1 shows two spectrograms of the synthesized waveforms and the chromagram for Bach's *Prelude and Fugue in C Major* (BWV 846). Spectrograms are diagrams showing spectra. Shades of blue correspond to low amplitudes; red marks medium amplitudes; yellow/green means maximum amplitudes. The code for generating the spectrograms is based on Dan Ellis's *Chroma Feature Analysis and Synthesis* tool¹ written in MATLAB.

¹<http://labrosa.ee.columbia.edu/matlab/chroma-ansyn/>

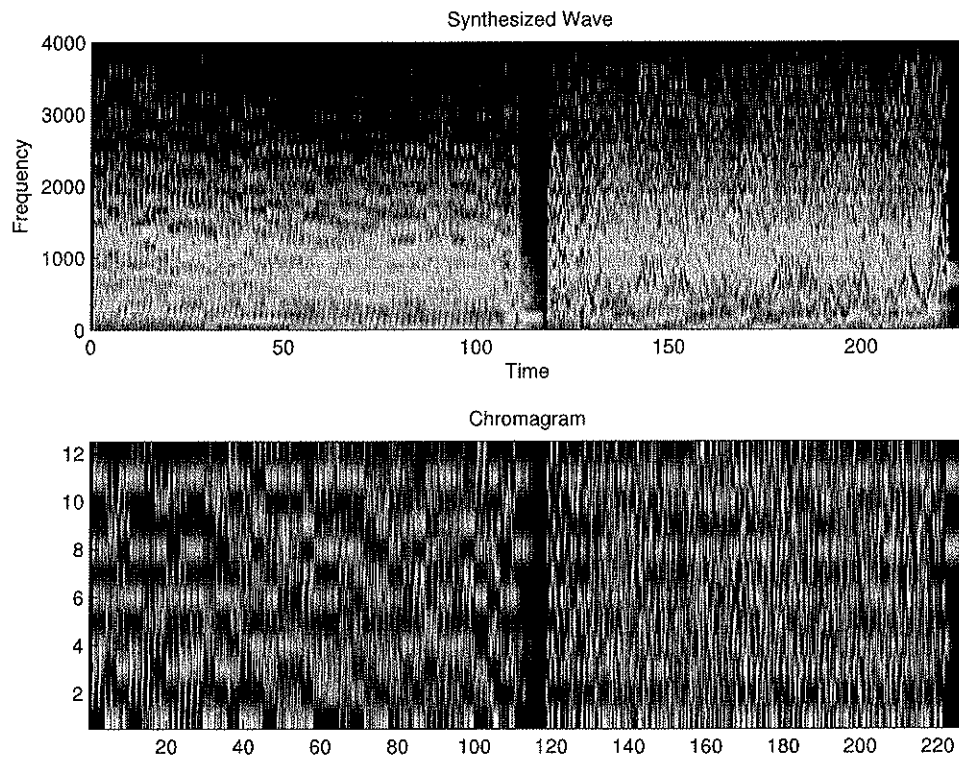


Figure 3.1: Synthesized waveform and chromagram for Bach's *Prelude and Fugue in C Major* (BWV 846)

Chapter 4

Hidden Markov Models

4.1 Markov Models

Before we discuss HMMs, let us first look at Markov Models in the context of chord progressions. In the following example, let's consider a chord progression with only three possible chords: CM, FM, and GM. We now set up a statistical model for predicting the n -th chord, c_n , based on the previous chords c_{n-1} , c_{n-2} , etc. More formally, we want to find the conditional probability

$$P(c_n | c_{n-1}, c_{n-2}, \dots, c_1), \quad (4.1)$$

the probability of the unknown n -th chord, $c_n \in \{CM, FM, GM\}$, depending on the known chord sequence $c_{n-1}, c_{n-2}, \dots, c_1$ in the past.

For example, if we know the last three chords are $\{FM, CM, FM\}$ chronologically, then the probability that the next chord would be GM is expressed as

$$P(c_4 = GM | c_1 = FM, c_2 = CM, c_3 = FM). \quad (4.2)$$

We can make inferences about this probability from the relative frequency of past observations of chord sequences.

Without further assumptions of how far in the past the n -th chord depends on, making inferences about c_n requires the frequencies for each of the three possible chords, for every previous time step, for a total of $3^{(n-1)}$ observations. Thus to fully consider past frequencies can quickly become infeasible. In light of this, we make the *first-order Markov assumption*:

$$P(c_n | c_{n-1}, c_{n-2}, \dots, c_1) = P(c_n | c_{n-1}) \quad (4.3)$$

"First-order" means the probability of a given value for c_n is assumed to depend only on the previous observation, c_{n-1} . A second-order Markov assumption would require both c_{n-1} and c_{n-2} .

Since our system assumes (4.3), our setup is a *(first-order) Markov model*. A chord sequence of our model is a *(first-order) Markov chain*, so-called because the probability of a specific chord sequence c_1, c_2, \dots, c_n is found by chaining the joint probability of each observation and

the previous one:

$$P(c_1, c_2, \dots, c_n) = \prod_{i=1}^n P(c_i | c_{i-1}). \quad (4.4)$$

With the first-order Markov assumption, the number of required past observations is significantly reduced. Instead of an exponentially growing numbers of statistics, we only need $3 \times 3 = 9$ observations to infer the probabilities of all possible sequences. $P(c_n | c_{n-1})$ is the same regardless of n , thanks to the Markov assumption, and $c_n, c_{n-1} \in CM, FM, GM$, therefore the 9 combinations.

For example, the probabilities $P(c_n | c_{n-1})$ may look like as shown in Figure 4.1. The arrows point from c_{n-1} to c_n , and the numbers next to them are the corresponding probabilities $P(c_n | c_{n-1})$. For instance, the diagram shows that $P(c_n = CM | c_{n-1} = GM) = 0.5$.

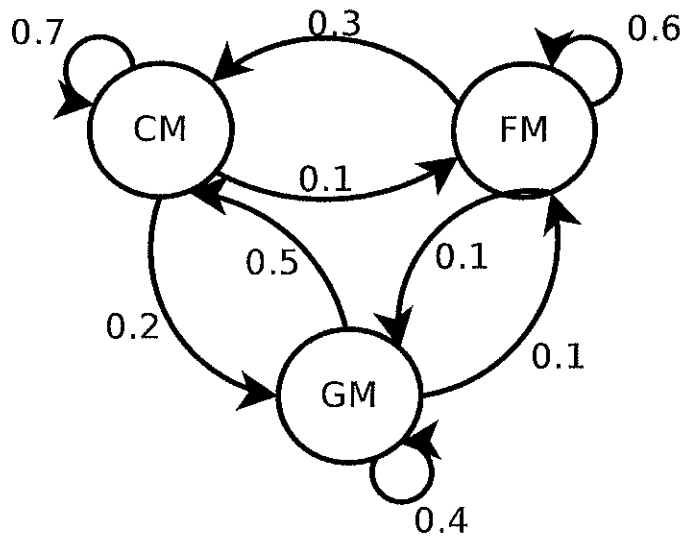


Figure 4.1: Example of a Markov model state transition graph

The above 9 probabilities are enough for us to infer the probability of all sequences, by simply multiplying all transition probabilities along the way.

4.2 Hidden Markov Models (HMMs)

Markov Models work well if we make predictions based on a set of states we can observe directly. In the previous example, we collect statistics of chords in the past, and directly predict what the next chord might be based on past observations. However, our project is about chord recognition from (artificially generated) acoustic information, and we do not observe what the chords are directly. Instead, our observations are acoustic signals, characterized by *feature vectors* (or *features*, see the next chapter). That is, we want to make inferences about chord sequences (hidden) based on features (direct observations). This is where *Hidden Markov Models* come in.

For simplicity, we assume for now that our observations are boolean (either true or false). Of course, actual feature vectors are much more complicated, capturing acoustic signals (like what our ears could hear). The boolean assumption helps us focus on the HMM itself, rather than faithful presentations of complex feature vectors at this point.

More specifically, the chord sequences are *hidden*. Finding what a certain chord c_n might be is based on only the observation at that time, namely feature vector f_i . Since f_i is assumed to be one-dimensional boolean, f_i is either *true* or *false*. That is, in this illustration, we have to find the probability of the current chord c_n , based on the acoustical observation f_i . Assume the probability of each of the three possible chords associated with a true value of f_i is according to the following table:

c_i	$f_i = true$
CM	0.8
FM	0.2
GM	0.5

Since

$$P(c_i|f_i) = \frac{P(f_i|c_i)P(c_i)}{P(f_i)} \quad (4.5)$$

by Bayes' rule, for a sequence of n chords $\{c_1, c_2, \dots, c_n\}$ together with a sequence of n observed features $\{f_1, f_2, \dots, f_n\}$, we have

$$P(c_1, c_2, \dots, c_n|f_1, f_2, \dots, f_n) = \frac{P(f_1, f_2, \dots, f_n|c_1, c_2, \dots, c_n)P(c_1, c_2, \dots, c_n)}{P(f_1, f_2, \dots, f_n)}, \quad (4.6)$$

where the probability $P(f_1, f_2, \dots, f_n|c_1, c_2, \dots, c_n)$ can be estimated as $\prod_{i=1}^n P(f_i|c_i)$, assuming independence among c -values and f -values.

Since our goal is to predict the chord sequence based on our observed features, we can safely omit the probability of the features alone, $P(f_1, f_2, \dots, f_n)$, because it is independent of the chords. However, omitting this term makes the result no longer an exact measurement for the probability, but rather a quantity directly proportional to the probability, which we refer to as the *likelihood* L :

$$P(c_1, c_2, \dots, c_n|f_1, f_2, \dots, f_n) \propto \quad (4.7)$$

$$L(c_1, c_2, \dots, c_n|f_1, f_2, \dots, f_n) = P(f_1, f_2, \dots, f_n|c_1, c_2, \dots, c_n)P(c_1, c_2, \dots, c_n). \quad (4.8)$$

Applying the first-order Markov assumption to two probability terms, we have

$$P(c_1, c_2, \dots, c_n|f_1, f_2, \dots, f_n) \propto \quad (4.9)$$

$$L(c_1, c_2, \dots, c_n|f_1, f_2, \dots, f_n) = \prod_{i=1}^n P(c_i|f_i) \prod_{i=1}^n P(c_i|c_{i-1}). \quad (4.10)$$

4.3 Formal Definition of HMMs

Before moving on to discussions of optimal pattern matching in HMMs, we need to specify terminology about various probabilities involved.

Our HMM involves the following quantities:

- The *set of states* $S = \{s_1, s_2, \dots, s_{N_s}\}$, where each s_i corresponds to one of the possible chords, and N_s is the number of states (chords).
- The *prior probabilities* $\pi_i = P(c_1 = s_i)$ are the probabilities of the chord s_i being the first chord of a sequence. In our model, we assume all chords are equally likely to be the first chord of a progression, so we have $\pi_i = \frac{1}{N_s}$ for all i .
- The *transition probability*, a_{ij} , is the probability to move from state i to state j , or equivalently, the probability chord s_i is followed by s_j in a certain progression. These probabilities are put together by the matrix A with entries

$$a_{ij} = P(c_{n+1} = s_j | c_n = s_i). \quad (4.11)$$

- The *emission probabilities* tell us how likely we will observe a certain feature vector f_k when we are currently at any specific chord (state). In our application, the feature vectors are discrete (thanks to mapping to twelve semitones, our features are twelve-dimensional binary vectors; see next chapter). In speech recognition, the observations are usually continuous, because the vocal frequency of the speaker can vary continuously over a wide range.

If K represents the total number of possible feature vectors, then $f_n \in \{v_1, v_2, \dots, v_K\}$ which is the set of all feature vectors. The emission probability $b_{ik} = P(f_n = v_k | c_n = s_i)$ is the probability of observing the feature v_k if the current chord is $c_n = s_i$. Similar to transition probabilities, the entries b_{ik} form a matrix B .

The above three parameters are referred to as the set of parameters $\Theta = \{\pi, A, B\}$ associated with the HMM.

In summary, our HMM works to match a (hidden) chord (state) sequence $C = \{c_1, c_2, \dots, c_N\}$ with the observation sequence $F = \{f_1, f_2, \dots, f_N\}$ of feature vectors.

4.4 Optimal Path

In our chord recognition problem, just like in any other form of pattern recognition, we want to find an optimal sequence of states (chords) that best match a sequence of observations (features). Specifically in chord recognition, we need to associate each frame of features to a certain chord, allowing us to locate the chord boundaries (where the current chord ends and the next chord starts). This process is referred to as *alignment* of acoustic features.

The word "optimal" is ambiguous; what is meant by "optimal" depends on the specific criterion. Here, our criterion for optimality is simply choosing the *path* (sequence of chords) that has the maximum likelihood according to the HMM. The recursive *Viterbi algorithm* can find this optimal path efficiently with a dynamic programming implementation.

The *Viterbi algorithm* works with two arrays of variables:

- $\delta_n(i)$, the maximum probability of a single path among all possible paths ending in chord c_i at time n :

$$\delta_n(i) = \max_{q_1, q_2, \dots, q_{n-1}} P(q_1, q_2, \dots, q_{n-1}, q_n = c_i, f_1, f_2, \dots, f_n | \Theta) \quad (4.12)$$

- $\phi_n(i)$, helping us keep track of the "optimal" path ending in chord c_i at time n :

$$\phi_n(i) = \arg \max_{q_1, q_2, \dots, q_{n-1}} P(q_1, q_2, \dots, q_{n-1}, q_n = c_i, f_1, f_2, \dots, f_n | \Theta) \quad (4.13)$$

4.5 The Viterbi Algorithm

In a nutshell, the Viterbi algorithm always finds the most likely path leading to each intermediate state (including the terminating state), and ignores all other paths. That is, only the *most likely* path leading to each chord c_i survives at each time n .

The **Viterbi Algorithm**:

1. Initialization

$$\delta_1(i) = \pi_i b_{i, f_1}, i = 1, \dots, N_s \quad (4.14)$$

$$\phi_1(i) = 0 \quad (4.15)$$

where π_i is the prior probability of being in state c_i at time $n = 1$.

2. Recursion

$$\delta_n(j) = \max_{1 \leq i \leq N_s} (\delta_{n-1}(i) \cdot a_{ij}) \cdot b_{j, f_n}, 2 \leq n \leq N, 1 \leq j \leq N_s \quad (4.16)$$

$$\phi_n(j) = \arg \max_{1 \leq i \leq N_s} (\delta_{n-1}(i) \cdot a_{ij}), 2 \leq n \leq N, 1 \leq j \leq N_s \quad (4.17)$$

To find the most likely path leading to a specific state j , it is enough to consider only those optimal paths leading to some previous state i , and the transition from i to j . The optimal substructure nature of this problem makes it perfect for an inductive dynamic programming implementation that runs in polynomial time, instead of a naive recursive implementation that runs in exponential time.

3. **Termination** When the last observation is reached, simply consider every state to be the terminating state, and find the maximum probability among all paths ending at each candidate.

4.6 Implementation

The system is implemented by using HMM Toolbox for MATLAB written by Kevin Murphy¹. The system is a 97-state HMM. One of the 97 states is for when there is no chord present in a frame window, and the other 96 states correspond to 96 chords. Since the root of a chord can be any one of the twelve semitones, and we are considering eight chord qualities as listed in Section 1.1, there are a total of $12 \times 8 = 96$ chords.

¹<http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html>

Chapter 5

Test Results and Conclusion

5.1 Testing

Test data consist of fifty selections. They went through the same chroma analysis as the training set did. The resulting chroma features are then fed into the trained HMM system. As the Viterbi algorithm finds the optimal path using the trained parameters and the input chroma features, recognition output is generated frame by frame. We then compare the recognized chords with the hand-labeled ground-truth (the true chord progression), and calculate frame-level accuracy of the system (percentage of frames where chords are correctly recognized).

5.2 Results and Analysis

When calculating the percentage of frames in which the recognized chords match the ground truth, we tolerated small discrepancies of chord boundaries by up to 5 frames (0.5 second). For example, if frames 20 to 36 are recognized as a G major triad, while the labeled truth says the same chord corresponds to frames 18 to 40, then frames 20 to 36 are all considered to be correctly recognized. On the other hand, if the labeled frame boundary for this chord is frames 18 to 30, then only 20 to 30 are considered correct while frames 31 to 36 are marked incorrect. This tolerance is reasonable since the ground truth was hand-labeled by listening to synthesized audio, so the chord boundaries can be slightly off by a few frames. In addition, chord boundaries can be interpreted in different ways in certain cases; for example, when two consecutive chords share a same note which connects the two, that note can be seen as the end of the first chord or the start of the second.

Figure 5.1 shows the recognized chord sequence of a phrase from Bach's *Prelude and Fugue in C Major (BWV 846)*. The segmented-lines show the chord names and boundaries recognized by the system, and the vertical lines correspond to ground-truth boundaries. In this selection, all frames are considered to be correctly recognized within our five-frame tolerance.

Table 5.1 shows frame-rate recognition accuracy in percentages. Analyses on the results show that most errors occur in selections with a faster tempo. Many of these errors are associated with non-chord passing tones. If the given input phrase has a fast tempo, each frame window will cover more notes; therefore, it is more likely that several non-chord tones,

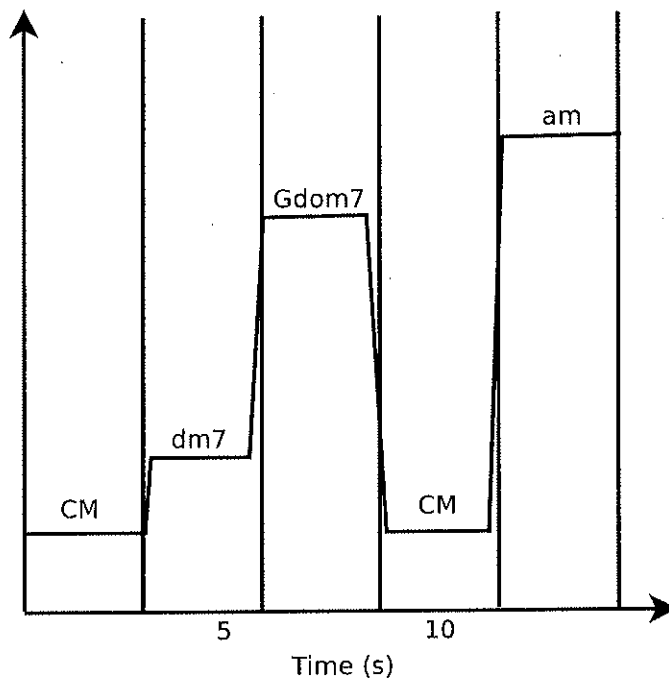


Figure 5.1: Frame-level chord sequence (state path) of a selection from Bach’s *Prelude in C Major (BWV 846)*. The vertical lines correspond to ground-truth boundaries.

or even two individual chords, are included in the same frame window. In these cases, the system has a much harder time figuring out the chords when many notes are clustered in a single frame.

This problem likely has to do with the fixed frame rate used in our system. A variable frame rate would help avoid this kind of problems, i.e. the faster the tempo, the smaller the frame window. However, an automated tempo identification requires beat identification. [9] suggests that beat-synchronous analysis as in [1] helps improve accuracy because the rate of chord changes is usually slower than beat changes, and non-chord tones usually occur off-beat.

Composer	# of Test Selections	Average Selection Length (second)	Accuracy
Bach	18	9.65	67.1%
Mozart	18	11.29	70.6%
Beethoven	14	12.08	64.8%
Overall	50	10.92	67.7%

Table 5.1: Frame-level accuracy on test selections

Another common type of errors has to do with confusion between non-chord tones and certain tones of a 7th chord. For example, E minor seventh chord consists of notes E, G,

B, and D. Three of these four notes - G, B, and D - also form G major triad. The system tends to mistakenly categorize E minor seventh as G major, if the note E does not appear strong in the feature vectors associated with the relevant frames. In many cases, the system correctly recognizes a 7th chord, but mistakenly moves to the related major triad mid-way before switching back to the correct 7th chord. We expect that this kind of confusion will be significantly reduced if the training data are greatly enlarged.

5.3 Conclusion

In this paper, we created an HMM-based chord recognition system trained on synthesized classical music selections. The overall accuracy of 67.7% is comparable to most systems in previous works. However, our system considers more chord qualities (diminished and augmented chords in addition to major and minor) than previous studies, as well as classifies 7th chords as separate categories. This shows that HMM-based systems perform nicely on classical music.

Our system is based on first-order HMMs, which only consider a single preceding chord when making decisions on the optimal path. Since music theory tells us there are some chord progressions more common than others, higher-order HMMs considering multiple preceding chords will most likely improve performance of the system. This will likely require a significantly larger training set, and may demand longer phrases or even entire compositions.

Bibliography

- [1] J. P. Bello and J. Pickens, "A robust mid-level representation for harmonic content in music signals," in *Proc. ISMIR*, 2005, pp. 304–311.
- [2] C. A. Harte and M. B. Sandler, "Automatic chord identification using a quantised chromagram," in *Proceedings of the Audio Engineering Society*, 2012.
- [3] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [4] A. Sheh and D. P. Ellis, "Chord segmentation and recognition using em-trained hidden markov models," *ISMIR 2003*, pp. 185–191, 2003.
- [5] D. Temperley, *The cognition of basic musical structures*. MIT press, 2004.
- [6] T. Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music," in *Proc. ICMC, 1999*, 1999, pp. 464–467.
- [7] M. Müller, *Information retrieval for music and motion*. Springer Heidelberg, 2007, vol. 6.
- [8] M. Müller and S. Ewert, "Chroma toolbox: Matlab implementations for extracting variants of chroma-based audio features," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 215–220.
- [9] K. Lee and M. Slaney, "Automatic chord recognition from audio using a supervised hmm trained with audio-from-symbolic data," in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*. ACM, 2006, pp. 11–20.
- [10] M. A. P. N. Perera and S. R. Kodithuwakku, "Music chord recognition using artificial neural networks," in *Proceedings of the International Conference on Information and Automation*, 2005, pp. 304–308.
- [11] S. Kostka and D. Payne, "Tonal harmony. new york: Alfred a," 2000.
- [12] C. Harte, M. Sandler, S. Abdallah, and E. Gómez, "Symbolic representation of musical chords: A proposed syntax for text annotations," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2005, pp. 66–71.
- [13] "Classical midi connection," <http://www.classicalmidiconnection.com>.

[14] "Classical piano midi page," <http://www.piano-midi.de/midicoll.htm>.

[15] "Timidity project," <http://timidity.sourceforge.net>.