

# CS 229 Final Project

## Decomposition of Musical Structure

Chuong Do, Sanders Chong\*  
*Stanford University*  
(Dated: March 21, 2003)

In this project, we examine the analysis of the musical form of a composition using a hybrid of unsupervised learning, support vector, and probabilistic learning techniques. Expectation maximization for Gaussian mixtures was used to identify common chord patterns in a database of MIDI files. We also demonstrate the utility of support vector classification of songs for identification of the tonic key, and finally employ a hidden Markov model based process for annotation of musical chord structure.

### 1. INTRODUCTION AND OVERVIEW

For artificial intelligence, music provides what some may consider to be an ideal test bed for machine learning systems. While inherently mathematical in its description, a musical composition is seldom so guileless in form that its structure may be deduced by simple inspection by the novice; decomposition of the structure of a piece is a complex task that takes experienced musicians years to master, involving the synthesis of knowledge regarding musical conventions and intuitive understanding of the composer's intentions in writing the song. Here, we present an automated learning method for elucidating the chord structure of a musical composition, in which we attempt to capture knowledge of common chord usages – by clustering together sets of notes in half-measures from a database of musical compositions – and intuitive understanding of chord progressions and flows in musical composition via a probabilistic model of musical form.

Machine learning approaches are no stranger to the field of understanding musical structure. Bayesian networks, neural networks, and linear classifiers have all been applied to learn musical properties such as style [1] and rhythm [2]. While machine learning techniques have been relatively successful at identifying and classifying key features in music, automatic accompaniment generation is still in the early stages of development.

Raphael modeled accompaniment beliefs using a Bayesian network with evidence propagation. However, his method required the manual setting of conditional distributions for accompaniment given soloist sections and hand crafted dynamics.

There has also been work on automatically generating music playlists [3]. SVMs with various Mercer kernels have been applied to this problem with some “successful” results. Some of the best results come from deriving a kernel from a set of meta-training functions that are related to a target function to be learned.

In sum, machine learning provides some very effective tools to identify and incorporate key music features in

learning applications. Bayesian belief networks, SVMs, and neural networks have all generated at least encouraging results. In this project, we further explore machine learning in classifying musical keys and chord types.

In particular, we describe methods for the automatic classification of musical compositions. Rather than attempt to handle such issues as music segmentation which have been explored by other groups in the past, we chose to focus specifically on the prediction of keys and chord structures, using MIDI files as the format of choice for analysis of tonal patterns in music. While our techniques were far from perfect, they did yield many useful lessons regarding computer understanding of music.

### 2. PRELIMINARIES

Before delving into the core of our paper it is important to establish a few key definitions crucial to understanding music decomposition:

1. Pitch: In general, pitch refers to the frequency of a note in a song. In our case, we chose to analyze tonal patterns without regarding to complex higher order musical structures, so we deal with only the twelve unique standard pitches of Western music (Ex: A, A#/Bb, B, C, C#/Db, D, D#/Eb, E, F, F#/Gb, G, G#/Ab).
2. Fundamental pitch: A song will typically be centered around one of the twelve pitches. We call this the fundamental pitch.
3. Mood: A piece may be classified as major or minor, which roughly correspond to notions of the song sound either “happy” or “sad”. Here, we distinguish between relative major and relative minor keys which are keys that have different base notes but are related in that they often involve the use of the same pitches in the song. Mathematically, a relative major and relative minor key are related in that the two have similar distributions of pitch usages in a song, but the fundamental pitch for the relative major key is 3 notes above that of the

relative minor key. The relative major or minor is represented as +1 or -1 as used later in our support vector classification.

4. Tonic: For major keys, we define the tonic to be the fundamental pitch, which is represented as an integer between 0 and 11, inclusive. For minor songs, we define the tonic to refer to the fundamental pitch of the relative major key corresponding to the minor key. Note that this working definition deviates from the traditional meaning of tonic as being essentially synonymous with fundamental pitch. As a further note, some songs do change tonic keys during the course of the music; for simplicity, we have chosen to disregard such cases.
5. Chord offset: Our goal in the project was to achieve some sort of automated understanding of patterns in music. Specifically, we chose to look at chord progressions, in which a chord is generally thought of as a particular group of notes that are commonly played together. To identify a particular chord, we use the chord offset to indicate the base note of a chord relative to the tonic. Since there are 12 different keys, there are  $N = 12$  possible chord offsets, representable as integers from 0 to 11, inclusive.
6. Chord type: Rather than use stock chord types, we decided to use chord profiles which give the relative intensity of pitches within a chord. The chord types are represented as a vector in  $\mathbb{R}^{12}$ , where each component represents the relative intensity (0.0 to 1.0) of a particular pitch relative to the base note of the chord. For the purposes of unsupervised learning (to be described later), the number of recognized chord types was arbitrarily set to  $K = 12$ .

For example, a G major triad played in a song in C minor may be represented as having a tonic of 3 (C), mood of -1 (minor), chord offset of 7 (interval between C and G), and chord type of  $([1.0 \ 0.0 \ 0.0 \ 0.0 \ 1.0 \ 0.0 \ 0.0 \ 1.0 \ 0.0 \ 0.0 \ 0.0 \ 0.0]^T)$

## 2. METHODS

Though the form of a musical piece can often be described by mathematical patterns, discovering the intentions of the composer automatically is a highly subjective process. Musical compositions contain more information than just a basic structure, and many forms may make use of the same notes. However, by selecting the appropriate music model and features, we may be able to apply machine learning techniques to aid us in our key-signature discovery. In this section we formulate the key-finding problem as a classification problem and

search for the maximum likelihood chord transitions beliefs.

### Step 1. Learning the Chords

Rather than develop an extensive database of preprogrammed known chords, we decided to learn the  $K$  chord types automatically by clustering known chords from a training set of MIDI files. As a simplification, we also decided to discretize our songs based on half-measures so that all notes within a half-measure are considered as occurring “simultaneously.”

Twenty sample MIDI files were hand-annotated, specifying the correct key signature, mood, and chord structure. Based on the annotation, all chords were translated so that the base note was zero (0). The profile for each half-measure is a vector in  $\mathbb{R}^{12}$  specifying the proportion of the measure for which any given pitch was active. A fundamental limitation of discretization is that it ignores the sequence of individual notes within each block.

In order to determine the  $K$  chord profiles, we used expectation maximization for Gaussian mixtures. Many resulting profiles corresponded nicely to known chord structures in music. In the examples shown below, the components of the 12-dimensional vector represent the probabilities of seeing certain pitches in a chord relative to the base note of the chord. For instance, a C major triad generally consists of the notes C, E, and G which are 0, 4, and 7 notes separated from the base note of the chord. Below are two of the clusters found using the mixture of Gaussians method.

- $[0.38 \ 0.00 \ 0.04 \ 0.00 \ 0.21 \ 0.02 \ 0.01 \ 0.25 \ 0.00 \ 0.05 \ 0.00 \ 0.04]^T \rightarrow$  major triad
- $[0.28 \ 0.02 \ 0.02 \ 0.26 \ 0.00 \ 0.03 \ 0.05 \ 0.25 \ 0.01 \ 0.02 \ 0.06 \ 0.01]^T \rightarrow$  minor triad

As a note, we also explored the use of the K-means clustering algorithm for chordal analysis which has the nice advantage of running much faster than EM. However, this approach tended to have the problem of decomposing large clusters of chords as having many different means rather than identifying unique clusters; the enforcement of a Gaussian distribution in the mixture of Gaussians model may simply correlate better to distributions of chord structures in music.

### Step 2. Decoding the Chord Sequence

Figuring out the chord progression of the song was accomplished using a fully connected  $N \times K$ -state first-order hidden Markov model (HMM). Each state corresponds to a particular chord offset and chord type. At first, one

might expect that chord types to be distributed independently with respect to the particular chord offsets; in actuality, given a particular tonic key, the various chord offsets are generally not equivalent in terms of their musical importance. As a result, once a tonic key for a piece has been specified, it is necessary to consider the sequence of both chord offsets and chord types as being pulled from a joint distribution.

Therefore, the state transition probabilities in the HMM correspond to probabilities of moving from particular combinations of chord offset and chord type from one half-measure to the next. The emission probabilities at each of these half-measures corresponds to the likelihood that the distribution of notes observed in the half-measure was generated by a particular chord offset and chord type pair. Given a particular tonic key, the HMM finds the maximum likelihood chord parse for the song.

Transition probabilities were estimated based on training set annotations. The annotations included only chord offsets per half-measure, since the chord types were automatically learned in Step One; pseudo-annotations were developed by matching each chord with the most similar chord type based on the chord offsets given in the actual annotations. An inherent problem with such a direct matching approach is that it does not incorporate knowledge of the transition probabilities between different chords into the initial training data. In practice, this works decently and was the only practical alternative to hand-classification of the samples.

One advantage of using the mixture of Gaussians approach instead of K-means clustering was that we were given covariance data for the identified clusters. For our purposes, then, we defined the probability of an emission as being the value of the multivariate Gaussian for a particular chord type given its mean and covariance matrix, multiplied by a scaling factor. The scaling factor may be found by evaluating the value of an emission in all K Gaussians corresponding to the different NK chord offset/chord type pairs and normalizing the sum of these values to be one.

The Viterbi algorithm was used to evaluate the log likelihood and recover the state sequence of the best possible chord parse for each of the 12 possible tonic keys. The maximum likelihood parse was then taken as the “correct” chord parse for the song.

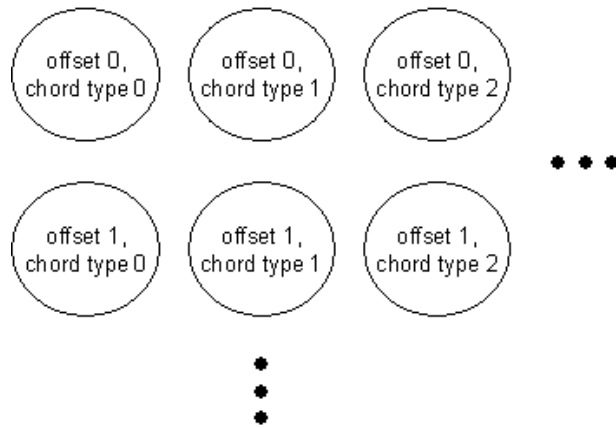


FIG. 1: HMM model

### Step 3. Key and Mood Identification

While the maximum likelihood chord parse is usually given by the correct relative major key for the song, these estimates are not always reliable, since the allowable chord types are not restricted to only major chords but rather are automatically determined by clustering.

Thus, for each of the twelve possible tonic (relative major) keys, we used a support vector classifier to determine whether the optimal chord sequence was generated using the given relative major key. Rather than training 12 different SVM models to recognize the 12 different relative major keys, we instead translated the song parse from each of the 12 different relative major keys to a A major. Since chord progressions relative to a tonic key never depend on the tonic key itself, then such processing is justifiable.

In most cases, only one of the support vector classifier runs is generally identified as being the correct key. In cases of ambiguity in the support vector classifier where more than one or zero keys were predicted, we picked the key given the most positive label by the support vector machine. This step fixes the key signature or tonic of the piece.

Once the most likely relative major key was identified, then we used another support vector classifier to determine, given the key signature and chord parse, whether the relative major or relative minor of the key signature is used. The reason why we chose to first identify the relative major key (tonic) and then whether the songs was major or minor rather than try to identify the fundamental pitch to begin with is that relative major and minor keys tend to have similar distributions of pitch usage even though they have different fundamental pitches. By focusing the efforts of the first classifier on picking the right tonic, the second classifier can then work on distinguishing between the nuances of the pitch distributions.

For each of the above steps, we encountered the problem of representing a chord parse in a vector that could

be used by the support vector classifier. On the intuition that chord transitions reveal more about the relative major or relative minor key than simply pitch distributions alone, we decided to process the chord parse to generate an  $NK \times NK$  boolean matrix  $S$  where  $S(i,j) = 1$  if and only if a chord transition from  $i$  to  $j$  occurs in the chord parse and 0 otherwise. Earlier, we explored setting the value of  $S(i,j)$  to instead be the proportion of the various chord transitions in the song but found that value to be too variable and our training set to be too small to generalize well. Using a hard classification for  $S(i,j)$  yielded decent prediction accuracy as described in the Results. In both of the above steps, training was performed using the pseudo-annotations for chord parses as described in Step Two.

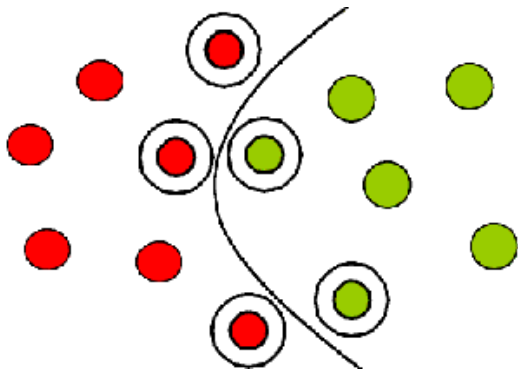


FIG. 2: SVM classification

#### Step 4. Playback

As the ultimate goal of the program was to perform automatic music classification, we decided to try a qualitative evaluation of the results by using the knowledge learned about the key/chord progression of a song to produce an accompaniment track that could be added to the MIDI. Essentially, this chord represents our best

guess of what chord an accompanist might play on top of the other notes in a given half-measure.

To do this, for each half-measure, we lookup the chord profile to find the likelihood that any chord may appear in the current half-measure. We then emit chords based on these probabilities. The most likely three notes are guaranteed to be played for the duration of the half-measure. Two additional notes are probabilistically selected to be played as quarter notes ( $\frac{1}{4}$  the time of a measure). All notes are added to a new MIDI track set to play on an acoustic grand piano.

### 3. RESULTS

In order to test our systems, we split the 20 hand-annotated MIDI songs into four groups for 4-fold cross validation. Output from our music decoder was compared directly to the labels we hand-annotated to yield the percent accuracy/error.

The program was generally successful in the prediction of the tonic and mood for most pieces. Songs in which the program had the most trouble tended to be ones in which the key was actually somewhat ambiguous to begin with (even for the human annotator!). However, chord level accuracy left much room for improvement. One explanation for this may be the fact that many different chords can be matched to a half-measure, partly due to ambiguity in music interpretations. Another possible reason is that the variance in possible chord types that were automatically trained may mean that some of the trained chord types were actually the result of training on mis-annotations in the original by the human annotations. Surprisingly, tonic and mood prediction were relatively insensitive to exact chord structure. As evidenced by the small difference in prediction accuracy between the training and test sets, generalization error for the chord structure prediction appears generally low, which is encouraging. Some statistics are given below:

Dataset	Size	Tonic Correct	Mood Correct	Number of Chords	Chords Correct
Test Set 1	5	4	4	790	524
Test Set 2	5	5	5	978	899
Test Set 3	5	5	4	924	757
Test Set 4	5	5	5	732	658
<b>TOTAL</b>	20	95%	90%	3424	83%
Train Set 1	15	15	13	2634	2031
Train Set 2	15	14	14	2446	1961
Train Set 3	15	15	14	2500	2150
Train Set 4	15	15	14	2692	2191
<b>TOTAL</b>	20	98%	92%	10272	81%

FIG. 3: Statistics

Qualitatively, the chords added to the new MIDI track are often the correct key and accompany the musical score very well. There are a few instances where it will play the relative minor chord when a major chord is expected (and vice versa). These occurrences, however, are in the minority so the general results are pleasing.

#### 4. FUTURE WORK AND CONCLUDING REMARKS

While our method was relatively successful in identifying the key signature of our sample music selections, there is still room for improvement. In particular we noted that there were some misclassifications between major and minor chords, perhaps due to insufficient major/minor-specific training data. Furthermore, some of the details within half-measures were lost due to our discretization. For more accurate, fine-detail note-emissions, it may be necessary to consider smaller blocks of time. Finally, our music-progression HMM may require extensions to include more music-specific information such as known styles, rhythms, chords, and music theory. Below are areas that we might consider if we were to extend our project.

- Chord type selection: Although we chose to automatically learn chord types via clustering with EM on Gaussian mixtures, it may have been more appropriate to stick with familiar chord types from music theory.
- Half-measure discretization: Dealing with music data in half-measure chunks was a simplification

that may have cost valuable information about the music track.

- Similarity metric for sequence profiles: Comparison of sequence profiles was based on rescaling the Gaussian mixtures fit during chord type selection; however, Euclidean distance is most likely not the optimal choice for measuring musical distance.

While not perfect for music decomposition, our methods do demonstrate the utility of machine learning approaches in understanding music.

---

\* Electronic address: [chuong.do@stanford.edu](mailto:chuong.do@stanford.edu), [chongs@stanford.edu](mailto:chongs@stanford.edu)

- [1] Dannenberg, R.B.; Thom, B.; and Watson, D. 1997. A machine learning approach to musical style recognition. In Proceedings of the 1997 ICMC. International Computer Music Association. <http://citeseer.nj.nec.com/dannenberg97machine.html>
- [2] Christopher Raphael. Automated Rhythm

- Transcription. In Proc. Int. Symposium on Music Inform. Retrieval. (ISMIR), pages 99-107, Bloomington, IN, USA, October 2001. <http://citeseer.nj.nec.com/article/raphael101automated.html>
- [3] Platt, John.; Burges, Christopher.; Swenson, Steven.; Weare, Christopher.; Zheng, Alice. Learning a Gaussian Process Prior for Automatically Generating Music Playlists. <http://research.microsoft.com/~jplatt/autoDJ.pdf>