Generative Models of Music

Mike Kayser
CS 229

Introduction

We propose a Markov-model based generative model for music. Latent variables in the model capture key and chord progression information. We train the model in a completely unsupervised setting and qualitatively investigate the accuracy of the latent variable assignments. We also quantitatively test the model on a composer identification task. Results indicate that the model can learn simple structural features such as key, but that more complex tasks such as chord recognition may require a more sophisticated model and/or the introduction of supervised training data. Both models are able to perform significantly better than chance on a composer identification task.

Motivation

Markov models have been successfully applied to a wide range of sequence analysis problems, e.g. in natural language processing. Generative models such as Hidden Markov Models are attractive because they can be easily utilized in a partially-labeled or unlabeled scenario.

This is welcome for two reasons. First is the usual reason that labeled data may be hard to come by and that there is far more unlabeled data (for example, in the form of raw MIDI or MP3 files) than labeled data available.

Second and somewhat more subtly, devising a reliable measure of "ground truth" for musical annotation may be a non-trivial task. For example, the same melody can often be harmonized in a variety of ways and there is often no single correct underlying chord progression. It may therefore be more reasonable to preserve uncertainty as a first-class feature and model *distributions* of chord progressions. Then, for example, if the task is to devise a harmonization for a given melody, one can sample from this distribution to obtain a valid harmonic progression.

Our goal in this work is devise a model which captures some of the latent structure of a piece of music. By "latent structure" we refer to things like key and chord progression. Our hypothesis is that models which recognize latent structure can be more accurate for a range of downstream tasks such as melody generation, harmonization, and composer identification.

Related work

Jan Buys[1] appears to have done similar work in generative modelling of chord and note sequences. We discovered his work too late to attempt any proper empirical comparison.

[1] http://www.cs.sun.ac.za/rw778/files/2011/11/j_buys_hons_report_2011.pdf

Models

We developed and trained two models, which we refer to as Model 1 and Model 2. All models are trained using the EM algorithm. First we will discuss some underlying modelling assumptions and simplifications common to both models.

Modelling assumptions

- We assume that two notes separated by one or more complete octaves represent the same pitch; that is, we collapse the set of all notes to a set of 12 *pitch equivalence classes*. For example, a high C note and a middle C note are treated identically.
- We assume that every musical piece has exactly one key which lasts for the entire piece. This is a strong assumption and not generally correct, but it makes our models somewhat more tractable. In the "Future Work" section we discuss potential methods of loosening this assumption.
- We assume that for every root pitch class, there are two types of keys: major and minor. We associate each of these two classes with a *set of pitch offsets from root* which classifies the "allowable" notes of this key. For example, the "major" key type allows notes which are 0, 2, 4, 5, 7, 9, or 11 semitones above the root key pitch. Note that "addition" in pitch values is *modular* addition, as implied by the first bulleted point above. See Figure 1 for examples of pitch class sets defined by keys.
- We therefore define a set of 24 possible keys: 12 major keys (one for each root pitch) and 12 minor keys. The "allowable" notes of any minor scale are defined by (what is music-theoretically known as) the "natural minor" scale; however, note that in general music theoretic terms the minor scale is more complex than this.
- We generate notes using a *mixture model*. For example, in Model 1 we generate a note by first generating a *note source* variable denoting whether the note was generated as part of the key or as part of a "background" distribution.
- When generating a note from a *key*, only pitches which belong to that key are allowed (all others are fixed at probability zero).
- The *background distribution* allows for passing tones and other non-key notes, and is fixed as a uniform distribution over all 12 pitch classes. In Model 2 we introduce chords as a third type of "note source." See the section "Model 2" below for more information.
- Notes are generated as *offsets* relative to a base pitch. For example, in Models 1 and 2, notes generated from key or from background are generated as offsets relative to the key's root pitch. This allows us to *share parameters*, e.g., to treat the tonic note in the key of C major in one training melody as the same event as the tonic note in the key of G major in another melody.

Key	**Allowable notes (pitch equivalence classes)**						
0 (C) minor	0	2	3	5	7	8	10
7 (G) major	7	9	11	0	2	4	6
5 (F) major	5	7	9	10	0	2	4

Figure 1: Selected examples of keys. There are 24 keys in total.

Model 1

Model 1 can be thought of as a simple "bag-of-notes" model with key as a latent variable. The generative story is as shown in Figure 2. Unless otherwise noted, all probability distributions are maximum-likelihood-estimated

multinomials. Note that as stated above, the probability distribution over notes is defined over *offsets* relative to the key's root pitch.

1. Generate a key root pitch $\mathbf{p_0}$ from {0,1,2,3,4,5,6,7,8,9,10,11}
2. Generate a key type **t** from {MAJOR,MINOR}
3. For each note of the piece:
    a. Generate the note source $\mathbf{s_i}$ from {KEY,BACKGROUND},
    b. If $\mathbf{s_i}$=KEY, generate $\mathbf{n_i}$ using the distribution $P(n_i\text{-}p_0 \mid s_i\text{=KEY}, t)$
    c. If $\mathbf{s_i}$=BACKGROUND, generate $\mathbf{n_i}$ using the distribution $P(n_i\text{-}p_0 \mid s_i\text{=BACKGROUND})$

Figure 2: Model 1 generative story.

Model 2

In Model 2, we introduce the notion of *chord progressions*. Specifically, a *chord bigram model* generates a chord sequence given a key type, and each chord generates an unordered bag of notes. An important assumption is that the "time granularity" of chord progressions is fixed; that is, a new chord is generated in sequence for each fixed-length time segment. In this case, the fixed *segment length* is defined in terms of fractions of a measure. In our experiments we generally assumed that there was one chord per measure.

Once the chord for a time segment is generated, the model generates all notes in the segment. Again we use a mixture model; in this case, there are three possible note sources: CHORD, KEY, and BACKGROUND. When generating notes from a chord, only notes consistent with the chord are allowed. Further, since in our model every chord consists of 4 notes, we share parameters across all chords for the first, second, third, and fourth note. That is, no matter what key or chord type, there are only four parameters to learn when generating notes directly from a chord. In Figure 3 below, $BIN(n_i,c_0)$ is a random variable drawn from {0,1,2,3} denoting which of the four chord notes is generated.

1. Generate a key root pitch $\mathbf{p_0}$ from {0,1,2,3,4,5,6,7,8,9,10,11}
2. Generate a key type **t** from {MAJOR,MINOR}
3. For each time segment of the piece:
    a. Generate the segment's chord root note $\mathbf{c_i}$, from {0,1,2,3,4,5,6,7,8,9,10,11}, using the distribution $P(c_i\text{-}p_0 \mid (c_{i-1}\text{-}p_0), t)$
    [At time=1, the history consists instead of a special START token]
    b. For each note in time segment **i**:
        i. Generate the note source $\mathbf{s_i}$ from {CHORD,KEY,BACKGROUND},
        ii. If $\mathbf{s_i}$=CHORD, generate $\mathbf{n_i}$ using the distribution $P(BIN(n_i,c_0)\mid s_i\text{=KEY})$
        iii. If $\mathbf{s_i}$=KEY, generate $\mathbf{n_i}$ using the distribution $P(n_i\text{-}p_0 \mid s_i\text{=KEY}, t)$
        iv. If $\mathbf{s_i}$=BACKGROUND, generate $\mathbf{n_i}$ using the distribution $P(n_i\text{-}p_0 \mid s_i\text{=BACKGROUND})$

Figure 3: Model 2 generative story.

Qualitative Investigation

Before testing the trained models on a downstream task, we performed a qualitative investigation to roughly determine the accuracy of the inferred latent structure.

We trained on the Bach Cello Suites, a selection of 36 pieces. We extracted note sequence information from freely available MIDI files. We trained models 1 and 2 and investigated their output to determine how accurately they determine (A) key, and (B) chord structure.

Key detection accuracy

We found that in both models 1 and 2, the detected key is almost exactly 50% likely to be correct. In fact, 100% of the time in our anecdotal observation, it was *either* the correct key or a closely related key, the "relative major" or "minor." This is because of a quirk in the music theoretic definition of "key" – that is, for any major key, there is a corresponding *relative minor key* which contains the same set of pitches. In practice, our model decides randomly whether to use a major key or its relative minor. We thought this ambiguity would be resolved by Model 2's notion of chord progressions, but in fact there is nothing strong enough in Model 2 to break the essential symmetry. One alternative would be to use priors to guide the model. For example, the *root chord* should have high probability in both major and minor keys. Such a prior would break the symmetry of the model and hopefully lead to better convergence.

Chord detection accuracy

Using Model 2, we find anecdotally that the chord detection accuracy is quite mixed. We did not perform a quantitative study due to a lack of annotated data, but it appears that accuracy in highly melody-driven, monophonic pieces like the Bach Cello Suites is below 50%. It seems there are a few reasons for the low accuracy:

1. The model changes chords too frequently, getting confused by "color" and passing notes. It may be that if notes were generated using an ordered Markov model such as a note bigram model, the model could learn that some notes "merely" serve as connectors between chordal notes, and thus that such notes need not be explained harmonically.
2. The model does not have an option to label an entire time segment as "no-chord."
3. The chord model is music-theoretically incomplete. In particular, due to the one-key-per-piece assumption, chords that belong to other keys, e.g. the dominant key, are not directly allowed for in the model. Bach, among many other classical composers, makes extensive use of modulation and tonicization (despite the fact that there is a natural "home" key for each piece); thus, the one-key-per-piece assumption is too strong.

Experiments

We tested the utility of the learned models on a composer identification task. In particular, we trained 4 different models on works by Rachmaninoff, Beethoven, Mozart, and Bach. (Overall, our data for training and test consisted of 46 works by Bach, 126 works by Beethoven, 50 works by Mozart, and 34 works by Rachmaninoff. We split data for each composer 50-50 between training and test.)

We used these trained models to assign probability scores to the held-out pieces of music. At test time, we choose a label for the piece of music based on the highest-scoring model. As shown in Figure 4 below, the model is able to perform at about 50% average accuracy, which is significantly better than chance. We are not aware of any experimental setups which are directly comparable to this, so it is difficult to know how this result compares to other modeling approaches.

The two models have significantly different composer-specific performances. Overall, it appears that Model 2 may indeed be slightly better, although it is difficult to say conclusively.

Composer	Model 1 Accuracy (% correctly classified)	Model 2 Accuracy
Bach	54.3	80.4
Mozart	44.0	42.0
Beethoven	38.9	51.6
Rachmaninoff	70.5	52.9

Figure 4: Model performance on composer ID task.

Future work

As mentioned above, there is much room for improvement in these models. These ideas can be grouped into thematic areas such as: greater supervision, more sophisticated generative models, and non-generative modeling approaches:

1. Improved supervision
    a. Prime the chord bigram model with a repository of known chord progressions. This can be done without specifically annotating the training MIDI files.
    b. Annotate some amount of data for hidden variables like key and chord progression.
2. More sophisticated generative model
    a. Develop a Markov model over keys.
    b. Allow notes to be generated by a note-bigram model rather than a bag-of-notes model. This might also involve introducing a new note source, “PASSING”, which allows the model to generate a note adjacent to the previous note.
    c. Use Bayesian methods to attempt to model “self-similarity” in music. For example, one might imagine a more sophisticated generative story in which a small vocabulary of “themes” is first generated, and then the note sequence is generated by a process which often generates transformations of the theme, and sometimes generates non-theme-related note sequences.
3. Use a different model. For example, it would be interesting to consider applying recurrent neural networks to note generation. One possible approach would be to divide a piece of music into very small “frames” similar to audio processing. A “frame” would be a vector (e.g., a 12-dimensional vector) denoting which notes are being played at a given very short time step. A RNN could be used as a continuous-state machine to model the dynamics of note frames. It would be interesting to investigate whether an RNN can recover latent properties such as chord progression and key.

Conclusions

We have developed a generative model of melodic and harmonic sequences. Our model gives what appears to be reasonable performance on a composer identification task, although there is much room for improvement. A qualitative investigation into latent variable assignment indicates that some combination of better modelling or more supervision may be necessary to correctly determine chord progressions.