Machine Learning in Automatic Music Chords Generation

Ziheng Chen
Department of Music
zihengc@stanford.edu

Jie Qi
Department of Electrical Engineering
qijie@stanford.edu

Yifei Zhou
Department of Statistics
yfzhou@stanford.edu

I. Introduction

Melodies and chords compose a basic music piece. Assigning chords for a music melody is an important step in music composition. This project is to apply machine learning techniques to assign chords to follow several measures of melody and generate pleasant music pieces.

In real music chord assignment process, to choose which chord to use for a measure, musicians normally consider the notes in this measure and how chords are progressed. So our aim is to learn the relationship between notes and chords, as well as the relationship between adjacent chords, and use that to assign chords for a new melody.

The input to our algorithm is a music piece with several measures. We then try different models to output a predicted chord for each measure. We will use basic models taught in the class (Logistic Regression, Naive Bayes, Support Vector Machine) as well as some advanced models (Random Forest, Boosting, Hidden Markov Model), and compare their performance for our problem.

II. Related Work

There have been some previous works using machine learning techniques to generate music chords. Cunha and Ramalho[1] set a neural network and combined it with a rule-based approach to select accompanying chords for a melody, while Legaspi et al.[2] use a genetic algorithm to build chords. Chuan and Chew[3] use a data-driven HMM combined with a series of musical rules to generate chord progression. Simon et al.[4] use HMM and 60 types of chords to made an interactive product, which would generate chord accompaniment for human voice input. Paiement et al.[5] use a multilevel graphical model to generate chord progressions for a given melody.

III. Dataset

A. Data Source

We collected 43 lead sheets as our dataset. The chosen lead sheets have several properties: a) virtually each measure in the dataset has and has only one chord; b) the chords in the dataset are mainly scale tone chords, which are the most basic and commonly used chords in music pieces. The data we use are in MusicXML format, which is a digital sheet music format for common Western music notation. Measure information can be extracted from this format by MATLAB.

B. Data Preprocessing

To simplify our further analysis, we did some preprocessing of the dataset before training:

1) The key of each song is shifted to key C. The key of a song can determine the note and a set of common chord types the song uses. A song written in one key can be easily shifted to another key by simply increasing or decreasing all the pitches in notes and chords equally, without affecting its subjective character. Therefore, without any loss of music information, we can shift the key of each song to key C to make the dataset more organized as well as decrease the number of class types (chord types) in training process.
2) The chord types (class types) are restricted only to scale tone chords in key C, which are 7 types: C major, D minor, E minor, F major, G major, A minor, and G dominant. Other types of chords (like C Dominant, G suspended, etc.) are transformed to their most similar scale tone chords.
3) Some measures in the dataset have no chords or no notes (like rest measure). We simply delete these measures from the dataset.
4) A small number of measures in our dataset are assigned two chords continuously to accompany

different notes in the measure. To simplify the learning process, we regard the second chord as the chord of this measure and delete the first one. From an audiences perspective, it will sound better than deleting the second chord.

After preprocessing, the 43 lead sheets we chose have 813 measures in total.

The 7 chord labels are shown in TABLE I. They are the labels we are trying to assign. Now, our project turns into a multi-class classification problem.

Label	1	2	3	4	5	6	7
Chord	C	Dm	Em	F	G	Am	G7

TABLE I: Labels for different chord types

IV. Feature Selection

We started by extracting some indicative features. The basic units for prediction are the notes inside a measure. Then we chose our initial features from the following three aspects:

1) Note itself: whether a note is present in the measure; take value of 0 or 1. A chord is influenced by the notes that appear to create a sense of harmony.
2) Note vs. Beat: the notes on the beat in the measure since chords should accompany the beat tones.
3) Note vs. Duration: the longest notes in the measure since the long notes need to be satisfied by the assigned chords.

To represent a note in a measure, we quantified it as shown in TABLE II. It is labeled 1 to 12. Note that we don't consider which octave the music note lies in, since the octave basically will not affect the chord type we choose (E.g. C4 and C5 are both labeled 1).

Label	1	2	3	4	5	6
Note	C	C#/Db	D	D#/Eb	E	F
Label	7	8	9	10	11	12
Note	F#/Gb	G	G#/Ab	A	A#/Bb	B

TABLE II: Labels for different note types

We chose notes on the 4/4 beats and 2 longest notes. In addition to the 12 type (1) features, we initially have 18 features.

To get the true effective features, we ran Forward Feature Selection based on Random Forest (which will be discussed further in the next section), by adding one feature at a time and selecting the current optimal one. The result is shown in Figure 1.

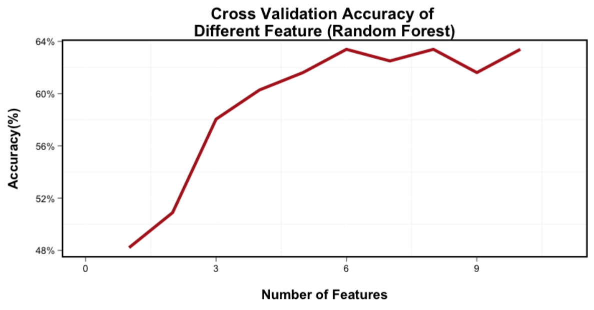


Fig. 1: Forward feature selection result

We can see the accuracy does not improve much after 6 features. Finally, we selected the next 6 features in our following analysis: the note pitches on the 4 beats and the 2 longest note pitches.

V. Models

A. Logistic Regression

We used multinomial logistic regression, i.e., softmax regression in R, with the log-likelihood as

$$\ell(\theta) = \sum_{i=1}^{m} \log \prod_{l=1}^{k} \left(\frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_l^T x^{(i)}}} \right)^{1\{y^{(i)}=l\}} \tag{1}$$

We could achieve the maximum likelihood estimate by using Newton's method:

$$\theta := \theta - H^{-1} \nabla_\theta \ell(\theta) \tag{2}$$

since our feature size is small ($n = 6$) and it is easy to compute the inverse of the Hessian.

B. Naive Bayes

We applied Naive Bayes by using the Statistics and Machine Learning Toolbox in MATLAB. We used the multinomial event model with multiple classes. For any class c, the maximum likelihood estimate gives

$$\phi_{k|y=c} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n_i} 1\{x_j^{(i)} = k \wedge y^{(i)} = c\}}{\sum_{i=1}^{m} 1\{y^{(i)} = c\} n_i} \tag{3}$$

$$\phi_{y=c} = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = c\}}{m} \tag{4}$$

We then made the prediction on the posterior computed with the probabilities above.

C. *Support Vector Machine (SVM)*

SVM is one of the supervised learning algorithms which is to maximize the distance between training example and classification hyperplane by finding (use hard-margin as an example):

$$\begin{aligned} \min_{\gamma,w,b} \ & \frac{1}{2}||w||^2 \\ \text{s.t. } & y^{(i)}(w^T x^{(i)} + b) \geq 1, i = 1, ..., m \end{aligned} \tag{5}$$

By using kernel trick, SVM can efficiently perform non-linear classification or data with high-dimensional features. In this problem, we tried the following three kernels in R:

1) Linear kernel

$$K(x_i, x_j) = \sum_{k=1}^{p} x_{ip} x_{jp} \tag{6}$$

2) Polynomial kernel

$$K(x_i, x_j) = \left(1 + \sum_{k=1}^{p} x_{ip} x_{jp}\right)^d \tag{7}$$

3) Radial kernel

$$K(x_i, x_j) = e^{-\gamma \sum_{k=1}^{p} (x_{ip} - x_{jp})^2} \tag{8}$$

D. *Random Forest*

Random forest is based on bagging which is a kind of decision tree with bootstrapping and can decrease variance. For a classification problems with p features, $\sqrt{p}$ features are used in each split in order to decrease the correlation of the trees[6]. We applied this model using R package.

After applying the model, we can generate the importance plot for the features as the figure below. All the features we are using have the similar importance level, which reinforces the conclusion from feature selection.

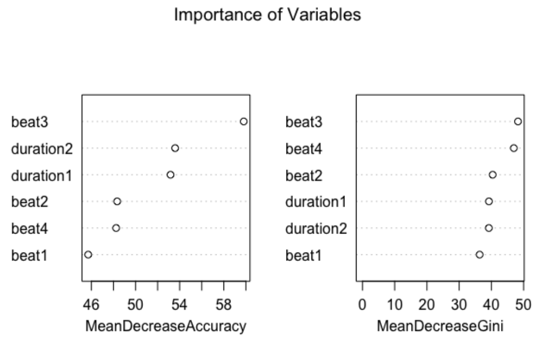


Fig. 2: Variable importance

E. *Boosting*

Boosting is a meta-algorithm to learn slowly to fit the new model from the residual of the current model. Its parameters include the number of trees to split, the shrinkage parameter and the depth of the tree[6]. We also applied this model in R.

Specifically, we tune these parameters using cross validation to make sure the model has the best performance. As a result, the number of trees is 200, the shrinkage parameter is 0.2 and the depth is 4.

F. *Hidden Markov Model (HMM)*

Up till now, these five models above make prediction based on the information of a single measure. We wanted to incorporate the relationship between measures. So, we also tried HMM to make prediction based on a sequence of measures.

In HMM, the system is being modeled to be a Markov process which has a series of observed states $x = \{x_1, x_2, ..., x_T\}$ and a series of related unobserved (hidden) states $z = \{z_1, z_2, ..., z_T\}$ (T is the number of the states). Suppose there are S types of observed states and Z types of hidden states. An $S \times S$ Transition Matrix denotes the transition probabilities between adjacent hidden states. And an $S \times Z$ Emission Matrix denotes the probabilities of each hidden state emit each observed state. Given an observed series of outputs, if we know the Transition Matrix and Emission Matrix, we can compute the most likely hidden series using the Viterbi Algorithm[7].

In our chord assignment problem, the first note of each measure was considered as an observed state, and the chord of each measure was considered as a hidden state. Using our dataset, we computed the transition probability and emission probability and formed transition matrix and emission matrix. And then we computed the most likely chord progression using `hmmviterbi` function in Statistics and Machine Learning Toolbox in MATLAB.

VI. Results & Analysis

A. *Cross Validation*

We started at comparing different machine learning models to indicate how they perform. We used hold-out cross validation and 30% of the data as the validation set. In addition, we also tried k-fold cross validation. The cross validation results are used to represent the test accuracy and evaluate how our models perform.

B. *Prediction on a Single Measure*

The cross validation accuracy for the five different models we used is shown in TABLE III. ($k = 5, 10$ for k-fold)

Model	70%/30%	10-fold	5-fold
Logistic Regression	43.75%	39.15%	38.42%
Naive Bayes	53.57%	55.34%	53.67%
SVM Linear	48.21%	36.97%	38.35%
SVM Poly	56.25%	52.80%	51.63%
SVM Radial	65.17%	61.39%	63.13%
Random Forest	62.50%	64.06%	63.73%
Boosting	63.39%	64.99%	62.29%

TABLE III: Accuracy for different models

To get an insight of the results above, we can have a look at the confusion matrix. Take SVM with radial kernel as an example:

$$\begin{bmatrix} 45 & 0 & 1 & 6 & 1 & 6 & 2 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 2 & 0 & 13 & 1 & 3 & 1 \\ 2 & 1 & 1 & 1 & 4 & 0 & 0 \\ 1 & 0 & 0 & 1 & 3 & 3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix} \tag{9}$$

In the confusion matrix, the rows represent the prediction results, the columns represent the true labels and the diagonal values are the correct predictions. We can see the data is highly imbalanced, i.e., most of the labels are 1 or 4. In fact, for key C, the frequent chords are exactly C major (1) and F major (4). This data distribution will cause our models to suffer.

Now let's visualize the accuracy results in the figure below.

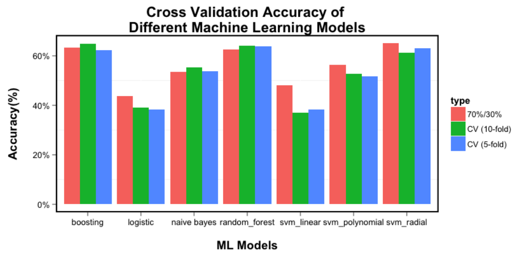
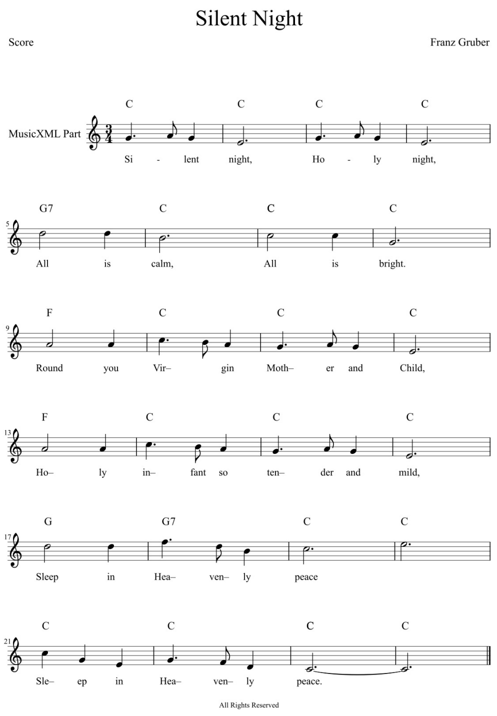


Fig. 3: Visualization of accuracy for different models

As the figure shows, random forest and boosting are the most solid models. Apart from the imbalanced data, logistic regression and SVM with linear kernel have bad performance, mainly because the relation is highly nonlinear and these models cannot fit this dataset while high bias/low variance classifiers (e.g., Naive Bayes) could have a better performance. However, Naive Bayes also assumes that the features are conditionally independent. In reality, the four notes on the beat could influence each other, so such strong assumption could limit the performance of Naive Bayes. Random forest and Boosting perform best in our case since they are both complex and have a reduction in model variance.

C. *Prediction on Sequential Measures*

The following is the song "silent night " that is assigned chords by our HMM model.



Silent Night
Score
Franz Gruber
MusicXML Part
C C C C
Si - lent night, Ho - ly night,
G7 C C C
All is calm, All is bright.
F C C C
Round you Vir– gin Moth– er and Child,
F C C C
Ho– ly in– fant so ten– der and mild,
G G7 C C
Sleep in Hea– ven– ly peace
C C C C
Sle– ep in Hea– ven– ly peace.
All Rights Reserved

Fig. 4: Generated lead sheet based on HMM predictions

The result we get for HMM varies greatly for different songs. The overall accuracy of HMM is 48.44% but for some pieces, it can achieve an accuracy over 70%. This could be caused by the limited information provided by the first note pitch observed. To add more pitches and regard a group of notes in the measure as an observation, the result can be improved but it will greatly complicate the model with a much larger state space.

The second reason is that, assigning chord is a more subjective than objective process. Two composers may choose different chord types for the same measure and both of them can sound pleasant. There is no single norm to decide if the chord is assigned correct or not.

VII. Conclusion & Future Work

From the results above, we can conclude that Random forest and Boosting perform best in prediction with single measure. HMM could also achieve a good result if we include more information as our observation states. But the highest accuracy we can achieve is only about 70%. This is caused by the subjectivity in our dataset.

This can be improved by choosing data from one composer with one genre. However, it is typically hard to achieve. A better way is to design an experiment to use human judgements to evaluate the performance of our model instead of only relying on the given labels.

References

[1] Cunha, U. S., & Ramalho, G. (1999). An intelligent hybrid model for chord prediction. Organised Sound, 4(02), 115-119.

[2] Legaspi, R., Hashimoto, Y., Moriyama, K., Kurihara, S., & Numao, M. (2007, January). Music compositional intelligence with an affective flavor. In Proceedings of the 12th international conference on Intelligent user interfaces (pp. 216-224). ACM.

[3] Chuan, C. H., & Chew, E. (2007, June). A hybrid system for automatic generation of style-specific accompaniment. In 4th Intl Joint Workshop on Computational Creativity.

[4] Simon, I., Morris, D., & Basu, S. (2008, April). MySong: automatic accompaniment generation for vocal melodies. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 725-734). ACM.

[5] Paiement, J. F., Eck, D., & Bengio, S. (2006). Probabilistic melodic harmonization. In Advances in Artificial Intelligence (pp. 218-229). Springer Berlin Heidelberg.

[6] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (p. 6). New York: springer.

[7] Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. Information Theory, IEEE Transactions on, 13(2), 260-269.