Machine Learning in Automatic Music Chords Generation

Ziheng Chen, Jie Qi, Yifei Zhou

Project TA: Albert Haque

Problem

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 piece.

Dataset

We collected 43 lead sheets to form our dataset. The data is in MusicXML format, which is a digital sheet music format for common Western music notation.

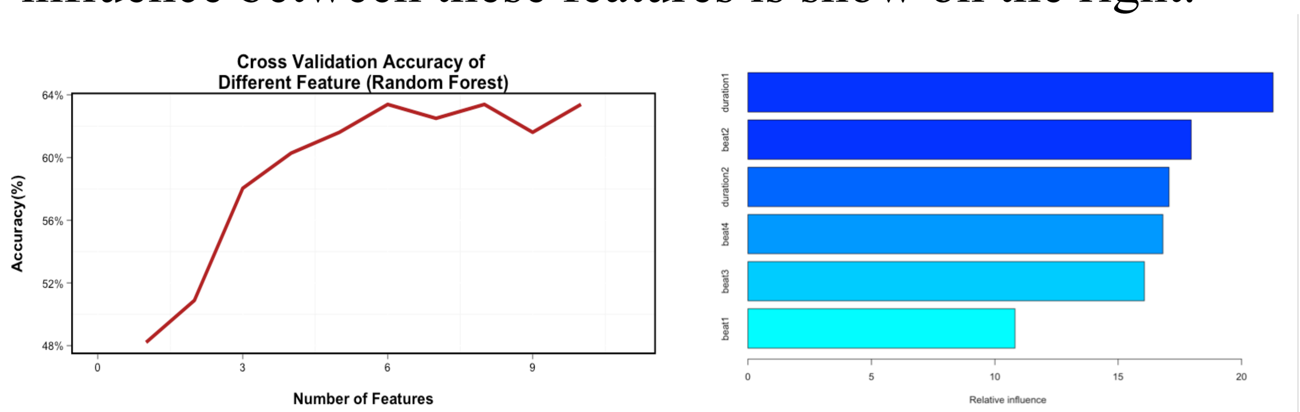
All the keys in the lead sheets are shifted to C. We deleted measures with no chords or no notes and chose the last chord if multiple chords are present in a measure. The chord types are restricted to 7 scale chord types in key C.

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

Feature Selection

We first extracted three kinds of features. (i) beat: the note pitch on measure beats; (ii) duration: the longest notes in the measure; (iii) note: if a measure contains a certain note.

Then we performed the forward feature selection and the result is show on the left. We selected six most indicative features: the four note pitches on the measure beats and the two longest duration notes in the measure. The relative influence between these features is show on the right.



For the note pitches, we quantified them into 12 values.

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

Models

- **Logistic Regression**

In our setting, we used multinomial logistic regression, i.e., softmax regression with the log-likelihood as

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

- **Naive Bayes**

We applied multinomial event model with multiple classes. For any class c, the maximum likelihood estimate gives

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

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

- **Support Vector Machine**

We tried SVM with the following kernels:

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

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

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

- **Random Forest[1]**

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.

- **Boosting[1]**

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.

- **Hidden Markov Model[2][3]**

We also tried HMM to incorporate the relationship between different measures, where the chords to be assigned are the hidden states and the first note pitches on the beat are the observed states.

Results & Analysis

- **Prediction on a Single Measure**

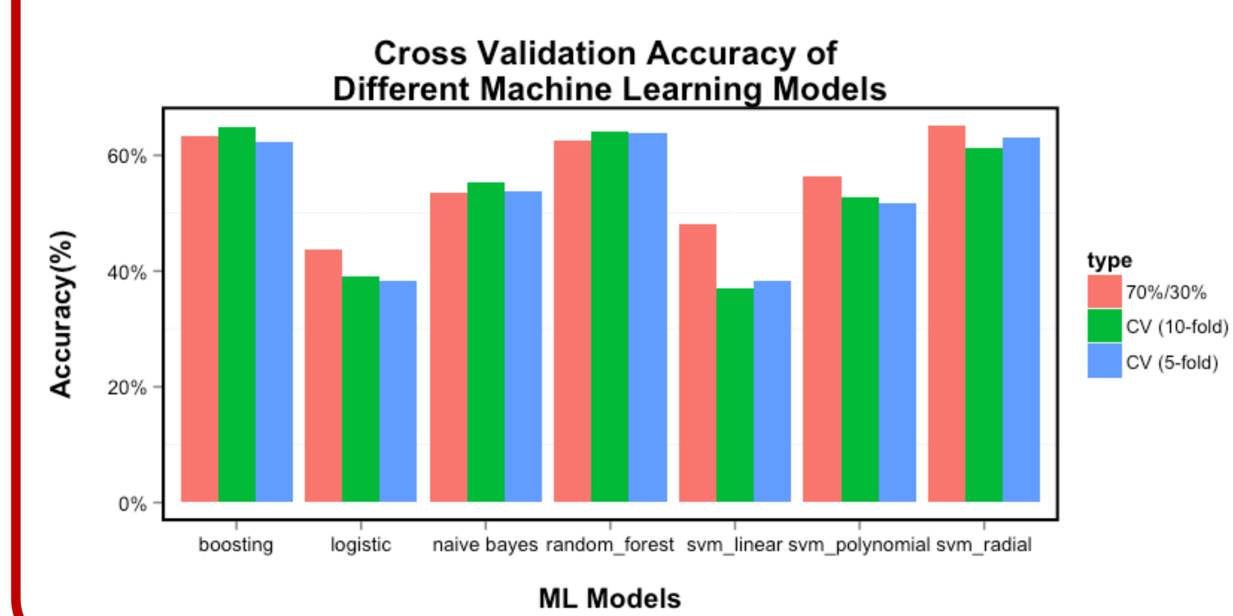
The first five models we used depend only on the current measure to make predictions.

For each model, it gives us a confusion matrix and the following is the one for SVM with radial kernel.

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

In the 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. In fact, for key C, the frequent chords are exactly 1 and 4.

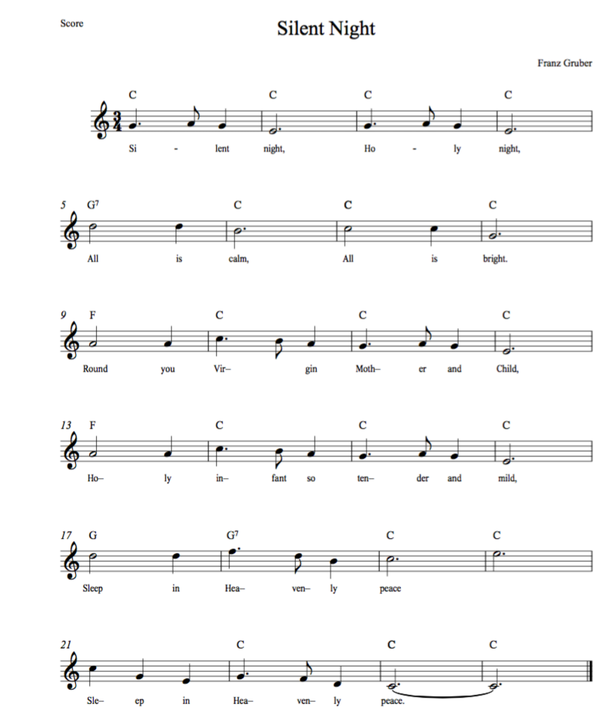
The cross validation accuracy for five different models we used is shown below.



As the plot shows, logistic regression and SVM with linear kernel have bad performance, mainly because the relation could be highly non-linear. Random forest and boosting are the most solid models since they are both complex and have a reduction in model variance.

- **Prediction on Sequential Measures**

Different from the previous five models, HMM uses a sequence of measures to make predictions. Here is an example of our prediction using HMM.



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 could improve it but will greatly complicate the model with larger state space.

Future Works

From the above results, we can see that the highest accuracy we can achieve is 65% with possible high bias. This is caused by the subjectivity in music pieces. For different composers, we could have different ways to assign chords. There is no absolute "correct" chord for a measure and the evolution methods can be improved.

For evaluation, we should use human's judgement to determine whether a chord is correct or not and design experiments to achieve this.

References

[1] G. James *et al.*, An Introduction to Statistical Learning with Application in R. Springer, 2013.

[2] Morris, Dan, Ian Simon, and Sumit Basu. "Exposing Parameters of a Trained Dynamic Model for Interactive Music Creation." *AAAI*. 2008.

[3] Simon, Ian, Dan Morris, and Sumit Basu. "MySong: automatic accompaniment generation for vocal melodies." *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2008.