A Machine Learning Approach to Support Music Creation by Musically Untrained People

Tetsuro Kitahara and Yuichi Tsuchiya
College of Humanities and Sciences, Nihon University
2-4-40 Sakurajosui, Setagaya-ku, Tokyo 156-8550, Japan
{kitahara,tsuchiya}@kthrlab.jp

Abstract

In this paper, we present an attempt of supporting music creation by musically untrained people. The main issues are (1) what human interface is suitable for such people to input their musical ideas and (2) how to generate musical pieces from their inputs that may be too abstract or musically incorrect. Machine learning technologies are expected to play a significant role for the second issue.

1 Introduction

Creating own musical pieces is one of the most wonderful ways to enjoy music. However, it is not easy especially for musically untrained people because creating musical pieces requires lots of musical skill and expert knowledge.

The development of computing technologies in recent dozens of years has made it much easier to create musical pieces. Software called MIDI sequencers and digital audio workstations (DAWs) has enabled us to create musical pieces on a computer. Typically, the user of such software inputs music data, such as a sequence of the pitches, onset times, and offset times of notes to be played, in a time-pitch plane displayed on the computer screen. It has made us free from the necessity of skill for playing an instrument. However, it still requires users to have an ability to imagine and express melodies that they want because they have to input them specifically.

About 30 years ago, Holland investigated how to use artificial intelligence technologies to encourage music composition by musically untrained people [1]. He attempted two different approaches. One is the Harmony Space, a 2-D visualization of the cognitive principle of harmony. On this visualization, users can compose a haromonic progression with simple manipulations. The other is a knowlege-based music tutoring system. This system focuses on helping users in how they should explore the Harmony Space.

Since his work, it has still been an open problem how to suppport music creation by musically untrained people. To achieve this, we have the following two issues:

Human interfaces Users need to be able to input their musical ideas in their mind in an easy, intuitive way. Data input by users should not be based on musical concepts that are difficult to understand (such as a traditional music notation).

Automatic music generation Users' inputs may be too abstract and/or musically incorrect to generate a musical piece. From such abstract, musically incorrect input data, the system needs to generate musically appropriate melodies with machine learning (ML) technologies.

Different easy-to-use human interfaces have been developed so far. Loop sequencers (e.g., Sony ACID Pro) enable users to create musical pieces only by combining short musical recordings prepared in advance. D-touch [2] is a tangible interface for music composition. Hyperscore [3] is an intuitive graphical interface based on freehand drawing for computer-assisted music composition.

In addition, there have been made a large number of attempts to generate musical pieces. Various ML techniques ranging from Markov models (e.g., [4, 5, 6]) to recurrent neural networks (RNNs) (e.g., [7, 8, 9, 10]) are used in these attempts. They will play an important role in achieving our goal.

In this paper, we present an attempt [11] from a series of our studies into support of music creation by musically untrained people [11, 12, 13, 14]. Suppose a situation that a user tried to obtain a melody with a full-automatic music composer[1] but the generated melody is not satisfactory. In such a situation, the user needs to be able to edit the melody in an intuitive, easy way. To achieve this, we focus on a continous curve called a *melodic outline* as a manipulation object. Given a melody generated by a full-automatic music composer, the system transforms it to a melodic outline. Then, the user can freely redraw this outline. After that, the system inversely transforms the outline to a melody. To transform the outline to a melody, we use a hidden Markov model (HMM).

2 Mutual transformation between melody and melodic outline

A melodic outline is a melody representation in which the melody is represented as a continuous curve. An example is shown in Figure 1. A melodic outline is mainly used for editing a melody with a three-step process: (1) the target melody, represented as a sequence of notes, is automatically transformed into a melodic outline, (2) the melodic outline is redrawn by the user, and (3) the redrawn outline is transformed into a note of sequence. The key technology for achieving this is the mutual transformation of a note-level melody representation and a melodic outline. Below, we present an overview of our method for this mutual transformation. See [11] for details.

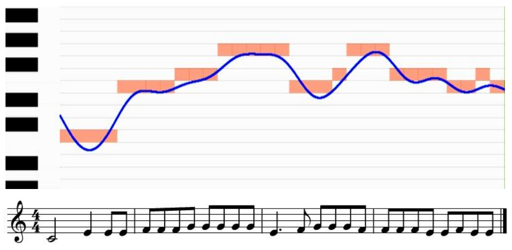
Figure 1: Example of melodic outline

2.1 Transformation of melody to melodic outline

The given MIDI sequence of a melody (Figure 2 (a)) is transformed into a pitch trajectory (Figure 2 (b)). The pitch is represented logarithmically. Regarding the pitch trajectory as a periodic signal, the Fourier transform is applied to this trajectory. Note that the input to the Fourier transform is not an audio signal, so the result does not represent a sound spectrum. Because the Fourier transform is applied to the pitch trajectory of a melody, the result represents the feature of temporal motion in the melody. Low-order Fourier coefficients represent slow motion in the melody while high-order Fourier coefficients represent fast motion. By extracting low-order Fourier coefficients and applying the inverse Fourier transform to them, a rough pitch contour of the melody, i.e., the melodic outline, is obtained (Figure 2 (c)).

2.2 Inverse transformation of melodic outline to melody

Once part of the melodic outline is redrawn, the redrawn outline is transformed into a note sequence.

First, the Fourier transform is applied to the redrawn outline (Figure 3 (a)). Then, the higher-order Fourier coefficients of the original pitch trajectory, which had been removed when the melodic outline is extracted, are added to the Fourier coefficients of the redrawn outline to generate the same pitch trajectory as the original melody from the non-redrawn part of the melodic outline. Next, the inverse Fourier transform is applied, producing the post-edit pitch trajectory (Figure 3 (b)).

Next, the pitch trajectory is transformed into a note sequence. An important thing here is to avoid notes that cause dissonance with the accompaniment. We achieve this using an HMM, which will be mentioned below. With the HMM, we obtain a sequence of post-editing note numbers[2]. The system outputs this in the MIDI format.

[1]Here, a *full*-automatic music composer means a system that generates a musical piece only by the user pressing the “Run” button (often with a few parameter inputs).

[2]Note numbers are integer values assigned to the pitches of notes. The central “C” pitch has a value of 60, and the value increases (or decreases) by 1 as the pitch gets one semi-tone higher (or lower).

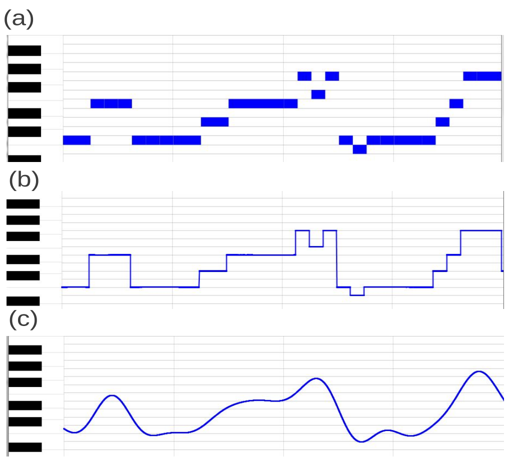

Figure 2: Flow of extracting melodic outline

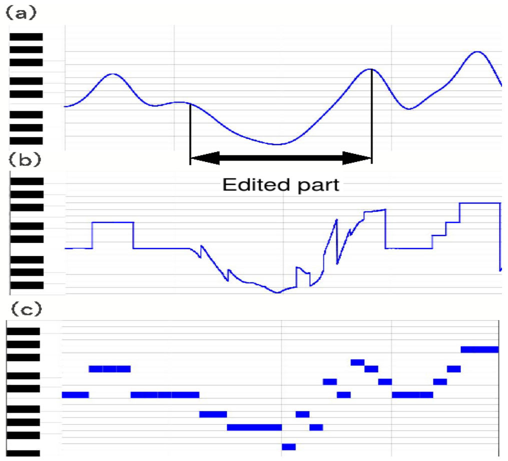

Figure 3: Flow of generating melody from outline

2.3 Hidden Makov model for generationg melody

The HMM presented here is designed to find a sequence of notes that has a reasonable trade-off between the closeness to the user's melodic outline and the musical appropriateness. The closeness to the outline is computed as the emission probabilities and the musical appropriateness is computed as the transition probabilities in this HMM.

The model is formulated based on the idea that the observed pitch trajcetory $O = o_1 o_2 \cdots o_N$ is emited with random deviation from a hidden sequence of note numbers $H = h_1 h_2 \cdots h_N$ that does not cause dissonance (the time resoluations of O and H are assumted to be equally set such as the eighth-note duration). What to do here is to estimate the most likely H (the most musically appropriate sequence of note numbers) from the given O (the pitch trajectory calculated based on the user-edited melodic outline).

Each hidden state s_i in this HMM correponds to the note number i. When the state of the observation at time t is s_i, it means that the note number i is chosen as the note to be output at time t (i.e., $h_t = i$). Because the emission probabilities model the closeness to the trajectory, they should be highest when the note number to be output is equal to the pitch of the trajectory. We therefore define the emission probability $P(o_t|s_i)$ for the state s_i as a normal distribution $N(i, \sigma^2)$, where the variance σ^2 is experimentally determined. In the current implementation, 36 states from s_{48} to s_{84} are used.

The transition probability $P(s_j|s_i)$ models the musical appropriateness of the note number j given the previous note i. Whether these transition probabilities are trained from data or manually set based on musical knowledge, the number of parameters should be less. We therefore simplify the transition probabilities as follows:

$$P(s_j|s_i) = p(\mathrm{mod}(j, 12))\, p_\Delta(\mathrm{mod}(|j-i|, 12)),$$

where $p(\cdot)$ is a unigram of octave-ignored notes and $p_\Delta(\cdot)$ is a unigram of octave-ignored note intervals. Ideally, these probabilities should be trained from actual music data, but for simplicity, they are manually fed in the current implementation as follows:

$$\text{Type I:}\; p(i) = \begin{cases} 16/45 & (i=0 & \text{(C))} \\ 2/45 & (i=2 & \text{(D))} \\ 8/45 & (i=4 & \text{(E))} \\ 3/45 & (i=5,9 & \text{(F, A))} \\ 12/45 & (i=7 & \text{(G))} \\ 1/45 & (i=11 & \text{(B))} \\ 0 & \text{(otherwise)} \end{cases} \quad \text{or} \quad \text{Type II:}\; p(i) = \begin{cases} 100/345 & (i=0 & \text{(C))} \\ 20/345 & (i=2 & \text{(D))} \\ 60/345 & (i=4 & \text{(E))} \\ 30/345 & (i=5,9 & \text{(F, A))} \\ 80/345 & (i=7 & \text{(G))} \\ 10/345 & (i=11 & \text{(B))} \\ 3/345 & \text{(otherwise)} \end{cases}$$

$$p_\Delta(d) = \begin{cases} 2/62 & (d=0 & \text{(perfect prime))} \\ 10/62 & (d=1,2,3,4 & \text{(min 2nd, maj 2nd, min 3rd, maj 3th))} \\ 6/62 & (d=5,7 & \text{(perfect 4th, perfect 5th))} \\ 1/62 & (d=6,8,10,11 & \text{(aug 4th, min 6th, min 7th, maj 7th))} \\ 4/62 & (d=9 & \text{(maj 6th))} \end{cases}$$

Above, $p(\cdot)$ and $p_\Delta(\cdot)$ are designed to generate typical melodies in the C major scale. Non-diatonic notes in the C major scale are strictly avoided for Type I while they may occasionally appear for Type II. These transition probabilities are applied only at each note boundary and no transitions are accepted between the onset and offset times of each note, because only pitch editing is currently supported for simplicity.

3 Experiments

We conducted several experiments on melody generation and system evaluation. Due to a lack of space, we here present only some examples of melody generation. See [11] and [13] for details.

Figure 4 shows examples of melody generation. Figure 4 (a) is an original melody generated by an existing automatic music composer [6]. Figure 4 (b) and (c) are melodies generated from melodic outlines edited by human subjects according to the instruction to make notes in the last measure lower. The notes in the last measure in the generated melodies have been actually made lower than the original ones. Figure 4 (d) and (e) are melodies generated from freely edited outlines. Above, Type II was used for $p(\cdot)$.

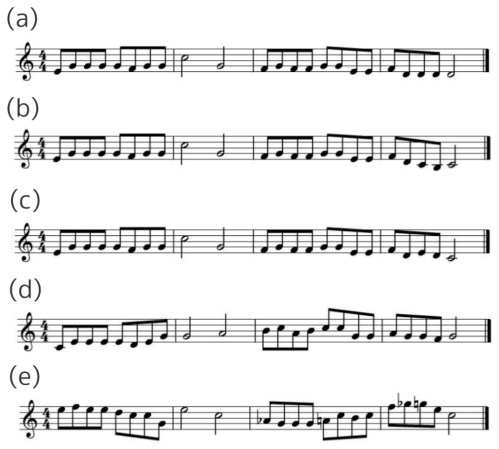

Figure 4: Results of melody generation

4 Discussion and Conclusion

Supporting musical creativity of musically untrained people is an interesting research topic from both human-computer interaction (HCI) and ML perspectives. From the HCI perspective, the central question is which abstraction level of human interface is suitable for manuplating musical content [15]. From the ML perspective, the main issue is to consider a trade-off between the closeness to users' inputs and the musical appropriateness. This is important because users' inputs are sometimes musically incorrect and may be even different from what they truly want. ML technologies play an important role in predicting what they truly want from their incorrect or too abstract inputs.

We used an HMM for this purpose. Although we omitted them, we obtained experimental results that participants at both novice and intermediate levels highly rated both the closeness to the drawn outlines and the satisfaction for generated melodies [13]. This implies that our system successfully supports their music creation to some extent.

However, we have many remaining issues as follows:

Data-driven model training We manually fed the HMM parameters to the system. Training the HMM from existing music data is an important next plan.

Style imitation It is possible to imitate a particular style of music (e.g., jazz) by training the HMM with music data of that style. If we prepare different HMMs corresponding to different styles of music, users can select and switch these styles according to their preferences.

User adaptation The balance between the closeness to the outline and the musical appropriateness can be controlled by changing σ^2. Because the best balance may be different for different users, this parameter should be adapted to each user.

More sophisticated ML models In recent years, ML technologies for generating time-series media have made remarkable progress including RNNs as mentioned in Introduction. We should introduce such state-of-the-art ML models.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers 26240025, 26280089, 16H01744, and 16K16180.

References

[1] Simon Holland. *Artificial Intelligence, Education, and Music: The Use of Artificial Intelligence to Encourage and Facilitate Music Composition by Novices*. PhD thesis, The Open University, 1989.

[2] E. Costanza, S. B. Shelly, and J. Robinson. Introducing audio d-touch: A tangible user interface for music composition and performance. In *Proc. DAFx 2003*, 2003.

[3] Morwaread M. Farbood, Egon Pasztor, and Kevin Jennings. Hyperscore: A graphical sketchpad for novice composers. *IEEE Computer Graphics and Applications*, 24(1):50–54, 2004.

[4] Francois Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*.

[5] ManYat Lo and Simon M. Lucas. Evolving musical sequences with N-gram based trainable fitness functions. In *Proc. IEEE Congress on Evolutionary Computation*, pages 601–608, 2006.

[6] Satoru Fukayama, Kei Nakatsuma, Shinji Sako, Takuya Nishimoto, and Shigeki Sagayama. Automatic song composition from the lyrics exploiting prosody of the Japanese language. In *Proc. Sound and Music Computing*, pages 299–302, 2010.

[7] Douglas Eck and Jurgen Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Proc. IEEE-NNSP 2012*, pages 747–756, 2002.

[8] Chun-Chi J. Chen and Risto Mikkulainen. Creating melodies with evolving recurrent neural networks. In *Proc. INNS-IEEE IJCNN 2011*, pages 2241–2246, 2001.

[9] I.-Ting Liu and Bhiksha Ramakrishnan. Bach in 2014: Music composition with recurrent neural network. arXiv:1412.3191, 2014.

[10] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proc. ICML 2012*, 2012.

[11] Yuichi Tsuchiya and Tetsuro Kitahara. Melodic outline extraction method for non-note-level melody editing. In *Proceedings of the Sound and Music Computing Conference 2013 (SMC 2013)*, pages 762–767, 2013.

[12] Tetsuro Kitahara, Syohei Kimura, Yuu Suzuki, and Tomofumi Suzuki. Hummi-com: Humming-based music composition system. In *ACM Multimedia 2012 (Technical Demo)*, pages 1321–1322, October 2012.

[13] Tetsuro Kitahara and Yuichi Tsuchiya. Short-term and long-term evaluations of melody editing method based on melodic outline. In *Proceedings of the Joint Conference of the 40th International Computer Music Conference (ICMC 2014) and the 11th Sound and Music Computing Conference (SMC 2014)*, pages 1204–1211, September 2014.

[14] Tetsuro Kitahara, Kosuke Iijima, Misaki Okada, Yuji Yamashita, and Ayaka Tsuruoka. A loop sequencer that selects music loops based on the degree of excitement. In *Proceedings of the 12th Sound and Music Computing Conference (SMC 2015)*, pages 435–438, July 2015.

[15] Haruhiro Katayose and Mitsuyo Hashida. Discussion on directability for generative music systems. In *SIG Technical Reports of IPSJ*, 2007-MUS-71, pages 99–104, 2007.