RL-MUSICIAN: A TOOL FOR MUSIC COMPOSITION WITH REINFORCEMENT LEARNING

A DRAFT

Nikolay Lysenko
nikolay.lysenko.1992@gmail.com

November 2, 2019

ABSTRACT

The notion of creativity is wider than generating something inspired by reference pieces of art. It also includes generation of something that meets criteria of being an art, but is not backed by existing pieces. On the one hand, supervised machine learning is limited by the former scope, because it requires a dataset. Reinforcement learning can be applied instead if the goal is to reveal more aspects of creativity. On the other hand, numerous recent advances in music generation are achieved with supervisedly trained neural networks. In this paper, a method that involves both reinforcement learning and neural networks is developed for automated music composition. An agent with shallow neural network as actor model is trained to compose musical pieces by interacting with a piano roll environment. The environment scores submitted pieces based on some hand-written evaluational rules derived from music theory. Cross-entropy method is used for training instead of gradient-based methods and so no datasets are needed.

Keywords: algorithmic composition · music generation · reinforcement learning

1 Introduction

Algorithmic music composition is automatic generation of outputs representing musical pieces and written in some formal notation. To name a few of common notations, there are sheet music, tablature, and MIDI standard. It is not required from output representation to unambiguously define sound waveform. For example, sheet music leaves exact loudness of played notes to discretion of a performer and may include only imprecise hints like pianissimo (very quiet). Anyway, there are parameters of sounds that must be determined by their representation. Usually, such parameters are pitch, start time, and duration.

Currently, there are no perfect tools for algorithmic composition. Plenty of various approaches for composing music automatically exist, but none of them produce well-structured and novel pieces that cannot be distinguished from works of a talented human composer. Thus, there is an open research problem. Several recent breakthroughs in it are accomplished with machine learning and both supervised learning [1, 2] and reinforcement learning [3] are applicable. Moreover, some researchers combine them [4, 5, 6].

In this paper, a new approach to music composition is suggested. Although it is quite straightforward, to the best of the author's knowledge it is not described anywhere, so the first contribution is its rigorous definition. The second contribution[1] is proper setup and tuning of the parameters. It is often the case that accurately tuned simple methods outperform complex methods [7] and, in addition, they are more transparent and less demanding.

Brief outline of the approach is as follows. There is an environment with 2D table representing piano roll which means that rows of the table correspond to notes and columns of the table correspond to time steps. A cell of the table contains 1 if the note is sounding at this step and 0 otherwise. An agent observes vector in which for every row there is an exponential moving average over all time steps up to the current time step (inclusively). An action of the agent is playing a note at the current time step. After certain number of actions are done, a movement to the next time step happens. When the last time step of the piano roll is reached, episode is ended and the agent receives reward based

[1] As of now, it is in progress.

on evaluation of created composition. Trainable parameters of the agent are weights of a neural network used as a so called actor model, i.e., a model that maps observation to probabilities of actions. These parameters are trained with cross-entropy method [8] (alternatively, they can be trained with genetic algorithms, ant colony algorithms, or evolutionary strategies [9]).

Obviously, the approach belongs to reinforcement learning, because there is an environment, but a dataset is absent. Nevertheless, there are strong connections with supervised deep learning. From generation of new pieces point of view, actor model returns probabilities of next notes given current state. This is similar to sampling new sequences from a next-step prediction model trained with maximum likelihood method on a dataset of existing sequences.

More details on the methodology are provided in Section 3, but here it is appropriate to discuss its advantages. The reasons for not involving supervised training at all are as follows:

- Finding new ways of music creation is a more challenging task than imitation of famous pieces. If no known pieces are used, chances are that the harder problem is considered and it is not replaced with the simpler problem of imitation.
- There are tuning systems other than equal temperament (for instance, in microtonal music). For some of them it may be impossible to collect dataset large enough to allow training models in a supervised fashion. However, developers of a tuning system should know some underlying principles and so (at least, in theory) it is possible to create evaluational rules and train an agent based on them.

Actually, results reported at this draft version are far from using above advantages at full scale. The current study is rather a proof-of-concept, but this proof-of-concept is easily extendable and ideas on how it can be improved are listed in Section 5.

2 Background and Related Work

2.1 Algorithmic Composition

Usage of computers for automatic composition of music dates back to 1950s and, what is more, approaches that can be labeled as algorithmic are known since Guido of Arezzo (11th century). Therefore, it is impossible to list here all studies about the subject. An interested reader can find more information in specialized reviews such as [10]. What is discussed below is just some examples related to reinforcement learning.

Reinforcement learning can be applied to algorithmic composition either as a standalone paradigm or in connection with deep learning. As for the first case, probably, there are no influential works and only separate studies like [11] are available. Conversely, the second group recently gained attention and several interesting approaches were developed there.

For example, reinforcement learning can be used for altering weights of recurrent neural networks trained to generate music sequentially [4, 5]. The goal is to make generated pieces more structured and conformed with music theory rules. To define an environment, let its state be composed of recurrent neural networks states and previously played notes, let an action be an output for current time step and let reward depend on both evaluational rules and probability of output according to initial RNN. Rewards are granted immediately after a step and so DQN (Deep Q-Networks [12]) are preferred as a training algorithm.

Usually, softmax activation function is used in the last layer of generative RNN. However, it is possible to generalize it with Boltzmann softmax which is a family of activation functions parametrized by one parameter called temperature and denoted as t. Depending on temperature, output distribution can vary from atomic distribution concentrated at the most probable action ($t = 0$) to distribution returned with softmax activation ($t = 1$) and to uniform distribution ($t \to +\infty$). In [6], Boltzmann softmax is used and also input vector is extended by introducing an additional part indicating origination of initial input (so called plan). An environment is defined so that state is a pair of temperature and plan, actions are changes in either temperature or plan, and reward depends on evaluation of produced with these settings piece.

Since algorithmic composition can be framed as training of a generative model, it sounds natural to try one of the most salient examples of generative models – generative adversarial networks (GANs) [13]. However, widespread notations for music assume sequences of discrete values but classical GANs work well only with continuous data, because gradient of discrete-valued functions is uninformative. Techniques originating from reinforcement learning can be used to overcome this obstacle [14, 15]. Namely, generator is trained with policy gradient method [16]. Such methodology is applied to various tasks and music composition is amongst them [17].

2.2 Cross-Entropy Method

Initially, cross-entropy method was developed for estimation of rare events probability [8]. However, it was found that it is also appropriate for solving optimization problems. More detailed discussion of cross-entropy method can be found in [18].

Algorithm 1 Cross-entropy method for optimization

Input: X – set of elements, $f : X \to \mathbb{R}$ – target function, $u(\cdot, w)$ – probabilistic distribution over X parametrized by vector w.
Output: $\hat{w}$ – approximate solution to the problem $\max_w \mathbb{E}_{x\sim u(\cdot,w)} f(x)$.
Hyperparameters: $w^{(0)}$ – initial value of w; N – number of iterations, n – number of vectors to draw at each iteration, σ – standard deviation for vectors generation, m – number of trials for each vector; ρ – fraction of best vectors to use for update; α – smoothing coefficient of updates.

for all $i \in \{1, \dots, N\}$ **do**
  for all $j \in \{1, \dots, n\}$ **do**
    draw $w^{(i,j)} \sim \mathcal{N}(\cdot|w^{(i-1)}, \sigma)$
    $r_j \leftarrow \sum_{k=1}^{m} f(x_k)$ where $x_k \sim u(\cdot, w^{(i,j)})$
  end for
  $r_{\text{threshold}} \leftarrow [\rho n]$-th highest value of $\{r_j : j \in \{1, \dots, n\}\}$
  $J \leftarrow \{j : r_j \geq r_{\text{threshold}}\}$
  $w^{(i)} \leftarrow \alpha w^{(i-1)} + (1 - \alpha)(\sum_{j\in J} w^{(i,j)})/[\rho n]$
end for
$\hat{w} \leftarrow w^{(N)}$

Algorithm 1 defines a variant of cross-entropy method for optimization. Sometimes, hyperparameter m is omitted and intermediate results are not aggregated over multiple trials. In case of $u(\cdot, w)$ that acts like a deterministic function of w, m is redundant, but, in general case, terminal result can be improved by setting $m > 1$.

In context of reinforcement learning, cross-entropy method is used for searching parameters of agent that result in maximum expected reward. Such approach is best suitable for environments where reward becomes known only after episode is finished. If there are rewards granted after intermediate steps, methods based on Q-values usually outperform it.

3 Methodology

3.1 Setup

For this study, piano roll is chosen as music representation format, but there are some subtleties.

Here, each row of a table does not correspond to a particular note or pitch. Relative axis is used instead of absolute axis. All that is important is that each row is one semitone higher than the row right below it. Thus, a musical piece is given only up to transposition and user-defined value for a pivot note is required to define absolute pitches. Similarly, duration of one time step is not defined by piano roll format and must be set externally too.

Another nuance is that there are no attempts to resolve ambiguity for successive cells of the same pitch. If values of 1 are contained by l successive cells of the same pitch, this line can be played either as one sound event that lasts l time steps or as l sound events that last one time step each or as k, $1 < k < l$, sound events of varying duration. Piano roll format does not specify it and evaluation of a roll is independent of ways to resolve it.

Denote number of rows as n and number of time steps as m and let piano roll be represented as matrix R of size $n \times m$. Suppose that current time step is t, $1 \leq t \leq m$. Then observation is defined as:

$$v = \sum_{i=1}^{t} \beta^{t-i} R_{\cdot i},$$

where β is a hyperparameter related to exponential decay, $0 < \beta < 1$.

It is important to distinguish between time steps of piano roll (columns of R) and time steps of an episode (moments when actions are taken by an agent). For a particular piano roll's time step t there are a episode time steps where a is a

hyperparameter. At each of these episode time steps, a cell from the t-th column may be filled with 1 or nothing may happen (if a cell that already contains 1 is chosen for action).

Agent takes action by drawing it from probability distribution returned by its actor model. An actor model can be any function that takes observation as input and returns probability distribution over cells from the current time step. Here, shallow neural network is used as actor model. Its weights can be viewed as something that defines transition matrix for a Markov chain from which sequence of piano roll's states is generated.

An episode starts with a piano roll that contains only zeros. Initial observation is zero vector of size $n \times 1$. An episode is being run until final piano roll's time step is reached. Then created piano roll is evaluated and resulting score is returned as reward for the episode.

3.2 Evaluational Rules

Eight properties of a final piano roll are evaluated at the current revision of the study.

- **Consonances.** Speaking formally, total reward for consonances is:

$$\sum_{t=1}^{m} \sum_{i,j=1}^{n} [i > j] R_{it} R_{jt} c((i-j) \bmod 12),$$

  where square brackets are Iverson brackets and c is a function that maps interval in semitones into measure of its consonance, $c : \{0, \ldots, 11\} \to \mathbb{R}$. The interpretation is as follows. Since $R_{it} \in \{0, 1\}$, only pairs of sounding cells from the same column affect the total sum. Value of each non-zero term depends on interval between the two corresponding pitches. The more consonant the interval is, the higher the value should be. Function c must be chosen in a way providing that. However, there is an exception – here, $c(0) = 0$ although intervals of whole number of octaves are perfectly consonant. This is done in order to prevent doubling the same melody in different octaves.

- **Absence of notes not from the specified scale.** Mode (major or minor) and position of a row with the tonic are set by a user before training. The same negative penalty is assigned to each sounding cell of a pitch that does not belong to the specified scale and then these penalties are summed.

- **Usage of the tonic.** A positive reward is assigned to a roll as a whole if share of columns where the lowest sounding note belongs to the tonic pitch class lies within a specified interval.

- **Conjunct motion.** A positive reward is granted for every sounding cell such that the previous cell of the same pitch is not sounding, but there is a sounding cell of close enough pitch in the previous column.

- **Contrary motion.** A positive reward is granted for every column such that there are both a sounding cell of pitch lower than that of the closest to it sounding cell from the previous column and a sounding cell of pitch higher than that of the closest to it sounding cell from the previous column.

- **Lines.** Ideally, a created composition should be decomposable into melodic lines. To have k melodic lines, it is necessary (but not sufficient) that exactly k cells are sounding in every column. So below penalty may be used as a proxy for scoring presence of desired number of melodic lines:

$$-\sum_{t=1}^{m} \left| k - \sum_{i=1}^{n} R_{it} \right|.$$

- **Noncyclicity.** Pieces where the same sequence of sound events is repeated over and over again should be penalized or, as a matter of choice, pieces should be rewarded for their noncyclicity. Here, the latter approach is used. The reward is implemented as minimum number of positive cells over differences between piano roll and its shifted version for all shift values from 1 up to a certain value (currently, it is equal to 8). To be more precise, this score is also clipped from above in order to prevent rewarding of placing extra sounding cells. In literature, scores based on autocorrelation are sometimes used as alternative way to push results away from degenerate states.

- **Absence of pitches played for a long time.** Partial case of a degenerate roll is a roll with some pitches that are constantly sounding. A negative penalty is assigned to every sounding cell such that cells of the same pitch are sounding in all l previous columns where l is a hyperparameter.

4 Experimental Results

A software implementation of the above methodology in Python programming language is available on GitHub[2]. The code has built-in documentation, is covered with unit tests, and is released as a package on PyPI[3]. All important settings are placed to a separate configuration file and so it is easy to experiment with them.

The implementation relies on some open-source tools [19, 20, 21, 22, 23].

To use less computation time, size of piano roll was chosen as follows: $n = 25$ (two octaves and one more note) and $m = 33$ (recall that the last time step is silent by design, so only 32 time steps are filled). Training an agent from scratch on a CPU of a regular laptop takes less than an hour with these settings.

As for human evaluation, created pieces lack rhythmic and tonal structure. Such drawback can be expected a priori, because there are no evaluational rules for structure-related properties. On the other hand, pieces are quite ear-pleasant. Thus, current version of the tool may be used by a human composer as an auxiliary source of inspiration. In particular, generated short parts further can be inserted into a longer piece.

5 Conclusion

This paper studies capabilities of reinforcement learning for music composition without usage of existing pieces. Here, music composition is framed as optimization problem where the goal is to find values of generative model's parameters maximizing average score of pieces generated with them. It is found that some progress can be made in this direction.

However, it is also found that vast majority of computational resources and engineering efforts are now spent on preventing an agent from taking actions that are discouraged by music theory. It is much easier to impose known constraints in program code than to learn them from environment responses. So it seems very promising to exclude discouraged actions and then focus training on things of higher level such as rhythm dynamic, presence of motiffs, proper final, and so on.

If setup becomes more complicated after above modifications, it might be worth trying to use LSTM neural network as actor model and evolutionary strategies as training algorithm.

References

[1] Daniel Johnson. Generating Polyphonic Music Using Tied Parallel Networks. In *International Conference on Evolutionary and Biologically Inspired Music and Art*, pages 128–143, 03 2017.

[2] Christine Payne. MuseNet. `openai.com/blog/musenet`, Apr 2019.

[3] B.D. Smith and G.E. Garnett. Reinforcement Learning and the Creative, Automated Music Improviser. *Evolutionary and Biologically Inspired Music, Sound, Art and Design. EvoMUSART. Lecture Notes in Computer Science*, 7247, 2012.

[4] Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. Generating Music by Fine-Tuning Recurrent Neural Networks with Reinforcement Learning. 2016.

[5] Nikhil Kotecha. Bach2Bach: Generating Music Using a Deep Reinforcement Learning Approach. *arXiv e-prints*, page arXiv:1812.01060, Dec 2018.

[6] Harish Kumar and Balaraman Ravindran. Polyphonic Music Composition with LSTM Neural Networks and Reinforcement Learning. *arXiv e-prints*, page arXiv:1902.01973, Feb 2019.

[7] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 101–109, 2019.

[8] Reuven Y. Rubinstein. Optimization of Computer Simulation Models with Rare Events. *European Journal of Operational Research*, 99(1):89–112, 1997.

[9] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *arXiv e-prints*, page arXiv:1703.03864, Mar 2017.

[10] Jose David Fernandez and Francisco Vico. AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research*, 48:13–582, Feb 2013.

[2] `https://github.com/Nikolay-Lysenko/rl-musician`

[3] `https://pypi.org/project/rl-musician/0.1.2/`

[11] Liangrong Yi and Judy Goldsmith. Automatic Generation of Four-Part Harmony. In *Fifth UAI Bayesian Modeling Applications Workshop*, Jan 2007.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv e-prints*, page arXiv:1312.5602, Dec 2013.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. 2014.

[14] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv e-prints*, page arXiv:1609.05473, Sep 2016.

[15] R Devon Hjelm, Athul Paul Jacob, Tong Che, Adam Trischler, Kyunghyun Cho, and Yoshua Bengio. Boundary-Seeking Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1702.08431, Feb 2017.

[16] Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.*, 8(3-4):229–256, May 1992.

[17] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv e-prints*, page arXiv:1705.10843, May 2017.

[18] P. T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A Tutorial on the Cross-Entropy Method. *Annals of Operations Research*, 134(1):19–67, 2005.

[19] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv e-prints*, page arXiv:1606.01540, Jun 2016.

[20] François Chollet et al. Keras. `https://keras.io`, 2015.

[21] Travis E. Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[22] Colin Raffel and Daniel P. W. Ellis. Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi. In *Music Information Retrieval Late Breaking and Demo Papers, 15th International Conference on*, 2014.

[23] Hao-Wen Dong, Wen-Yi Hsiao, and Yi-Hsuan Yang. Pypianoroll: Open Source Python Package for Handling Multitrack Pianoroll. In *Soc. Music Information Retrieval Conf.*, 2018.