EMERGENT PATTERNS: MUSIC GENERATION THROUGH MARKOVIAN FEEDBACK

|  |  |  |
| --- | --- | --- |
|  | Felipe Tovar-Henao |  |
|  | University of Cincinnati College-Conservatory of Musictovarhfe@ucmail.uc.edu |  |

ABSTRACT

*This paper presents a feedback-based implementation of Markov decision processes for symbolic music generation, aimed at addressing a well-known trade-off between novelty and coherence in regular nth-order Markov chains — novelty used here to denote the degree to which the output sequences deviate from the input sequences, and coherence defined as the long-term presence and persistence of patterns in the output sequence. The simplicity of Markov models, relative to state-of-the-art deep-learning architectures, serves as a framework for exploring potentially transferable and scalable solutions to the perennial problem of preserving complex, long-term musical dependencies in generative music.*

1. INTRODUCTION

Recent breakthroughs in artificial intelligence have caused a dramatic shift in generative music research, favoring the use of deep learning architectures, such as transformer models [1], over more traditional, rule-based approaches, including cellular automata, constraint satisfaction problems, and Markov chains [2]. Notable examples of state-of-the art (SOTA) generative music models include [3], [4], [5], capable of directly generating raw audio data given a text prompt. Despite these nontrivial improvements, one of the remaining problems in generative music is that of preserving complex, long-term musical dependencies in the output [6], [7]. In other words, maintaining pattern relationships over long spans of time.

To explore this issue, this paper considers a deliberately simpler case, namely Markov decision processes, as a framework for exploring potentially transferable solutions to more complex cases, such as deep learning architectures. In particular, it introduces a web-based feedback-based implementation of Markov models for symbolic music generation in which long-term dependencies emerge in the generated output, without durational constraints.[[1]](#footnote-1)

* 1. Markov models

Markov models and variants thereof have traditionally been used as an analysis/resynthesis approach to symbolic music generation [8], [9], [10]. This consists of generating *nth*-order probabilistic models of pre-existing musical data, usually in MIDI format, from which new musical sequences can be generated through Markovian predictions. These models are formalized as finite-state transition probability matrices, describing the probability of every state transitioning into all other possible states (see Figure 1).

*Copyright: © 2024 Tovar-Henao. This is an open-access article dis- tributed under the terms of the* [*Creative Commons Attribution License 3.0 Unported*](http://creativecommons.org/licenses/by/3.0/)*, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.*

|  |  | **FUTURE**  (output) | | | |
| --- | --- | --- | --- | --- | --- |
|  |  | *state 1* | *state 2* | *…* | *state* n |
| **PRESENT**  (input) | *state 1* | 0.1 | 0.2 | … | 0.7 |
| *state 2* | 0.6 | 0.3 | … | 0.1 |
| *…* | … | … | … | … |
| *state* n | 0.3 | 0.2 | … | 0.5 |
|  |  | (Transition probabilities) | | | |

**Figure 1**. A transition probability matrix, where each row represents the normalized probabilities of the present state transitioning into all other possible future states.

In the context of symbolic music generation, a Markov state often stands for a musical event, such that new musical sequences can be stochastically generated by repeatedly predicting the next state or musical event, using the *n* most recent predictions as input. However, *nth*-order Markov models suffer from a well-known tradeoff with respect to their order — namely that lower-order Markov models generate sequences with long-term dependency problems (i.e., pattern-lacking), while higher-order Markov models result in sequences too similar to the input data. This can be understood as a *novelty-coherence* tradeoff— *novelty* used here to denote the degree to which the output sequences deviate from the input sequences (i.e., dissimilarity), and *coherence* defined as the long-term presence and persistence of patterns in the output sequence.

In light of this limitation, a feedback-based implementation of a Markov model is presented here, which affords the generation of polyphonic music sequences with high levels of both novelty and coherence, as defined previously, relative to regular *nth*-order Markov models.

Diagram

Description automatically generated

**Figure 2**. Piano roll visualization of a 5-stage order-boosting feedback, using J.S. Bach’s *Invention No. 4 in D minor* (BWV 775) as input data. Sequential segments that repeat at least once within each sequence are highlighted and differentiated by label and enclosing shape.

* 1. Background

The implementation presented here is motivated by two common approaches in machine learning: *reinforcement* *learning* and *diffusion models*, both of which are based on Markov processes.

* + 1. Reinforcement learning

This paradigm, popular for being used to train computer agents how to play games, consists of cumulatively rewarding an agent for taking actions that lead to a desired outcome, thus increasing the probability that the system will make optimal, or nearly optimal decisions in future opportunities. While establishing a desired outcome in the context of music generation is a highly subjective, and thus limiting approach, reinforcement techniques can still be a useful mechanism to increase the probability of repeated transitions between Markovian states, which in turn increases the chances of pattern emergence in a musical sequence.

* + 1. Diffusion models

Diffusion probabilistic models are a type of deep-learning architecture that is commonly used for image generation tasks [11]. During the training process, these models gradually apply Gaussian noise to the input data over multiple timesteps, and then learn how to remove the noise iteratively. Since the state of the data at any given timestep is only dependent on the previous state, the data noising and denoising stages can be described or modeled as Markov processes.

This idea of generating data by way of iteratively removing or sculpting noise, comparable to subtractive audio synthesis techniques, motivates the approach taken in this implementation.

* 1. Method

The web-based implementation presented here makes use of two relatively simple but core mechanisms, here referred to as *order-boosting feedback* and *prediction reinforcement*. These are inspired by *diffusion models* and *reinforcement learning*, respectively.

* + 1. Order-boosting feedback

This consists of first building a low-order transition probability matrix, typically of 1st-order, based on the original input sequence, and then generating a new sequence from it. This new sequence can then be used as input or training data to repeat the same process but increasing the Markov order by 1 with each iteration. The lower the order of the initial Markov model, the noisier or more pattern-lacking the first generated sequence is — i.e., more *novel*. A possible interpretation of what this feedback process does is the promotion of *pattern emergence*, whereby repetitions of increasingly larger segments start to the emerge from the “musical noise” generated by the initial lower-order Markov model. This process is shown in Figure 2, using J.S. Bach’s *Invention No. 4 in D minor* (BWV 775)as the initial input sequence. All intermediate generated sequences, each around 150-notes long, are also included.

In the first iteration, the generated sequence is noisy, with a short pattern that only repeats once, highlighted with an oval-shaped enclosing. With each subsequent iteration, more and larger pattern repetitions begin to emerge, until the final sequence contains mostly repeated patterns. It’s also worth noting how these patterns seem to gradually converge into meso-formal sections whereby, in the fifth iteration, segments A and B appear to blend into a single section, while segment C stands on its own, thus resulting in a ternary structure. Crucially, the final sequence contains novel patterns relative to Bach’s *Invention* *No. 4* and it’s able to maintain long-term dependencies.

* + 1. Prediction reinforcement

Prediction reinforcement is used to continuously update the transition probability matrix as the sequence is being generated. It consists of indiscriminately rewarding every new Markovian prediction, by increasing the transition probability between the current state and the predicted state. The transition probability of the predicted state is thus multiplied by a *reinforcement factor*, and then normalized along with all transition probabilities for the current state such their sum is equal to 1. The matrix is updated with every new prediction, so as to incentivize repetitions between transitions throughout the output sequence. However, since this may result in undesirably high repetitions, a *maximum* *reinforcement threshold* can be set, which ensures that reinforced transition probabilities are clamped when such threshold is exceeded. This reinforcement step can be expressed mathematically as:

|  |  |
| --- | --- |
| ) | (1) |

where *M* is the transition probability matrix, *i* and *j* are, respectively, the row and column indices of the prediction’s transition probability, *N* is the number of matrix columns, *k* is the reinforcement factor, and *h* the maximum reinforcement threshold, in the [0, 1) interval.

Rectangle

Description automatically generated with medium confidence

**Figure 3**. Two sequences generated by a 1st-order Markov model, based on the clarinet part of Mozart’s *Clarinet Concerto in A KV. 622* (Adagio). The first with no reinforcement, and the second with a *reinforcement factor* of 2 and no *maximum reinforcement threshold*.

An extreme case of this reinforcement mechanism’s impact on the output can be visualized in Figure 3, where two sequences, both generated by a 1st-order Markov model, are compared: the first with no reinforcement, and the second with a reinforcement factor of 2 and no maximum reinforcement threshold. The result is a lack of repetition in the first sequence, and the second sequence quickly locking into a loop. While this degree of repetition is generally undesirable, using more moderate values for the reinforcement factor and maximum reinforcement threshold can prove useful in promoting pattern emergence when combined with order-boosting feedback.

* 1. Implementation

A visual representation of the implementation, including the preprocessing stage and the two previously described mechanisms, is shown in Figure 4. The reader is encouraged to experiment with the web-based implementation[[2]](#footnote-2).

Graphical user interface, qr code

Description automatically generated with medium confidence

**Figure 4**. Visual representation of Markovian feedback generation.

* 1. Sequence generation

18 different musical sequences were generated using the current implementation. Details related to the training and generated MIDI files, as well as the input parameters used, are included as part of the paper’s supplementary files.

* + 1. Training data

The data used as input to generate the examples consisted of five, well-known piano works from the Western Classical repertoire, in MIDI format. The motivation behind this choice was to facilitate the perceptual comparison between the generated sequences and music that the reader would likely be familiar with.

* + 1. Preprocessing

The preprocessing stage consisted of performing the following operations on the input MIDI files:

1. *Sorted array representation:* For each MIDI file, all note events were sorted by the following 3 features, in order of priority: (1) start time, (2) pitch, and (3) instrument MIDI number. Each MIDI file would then be represented as an array of note events and, where multiple files were used to generate a sequence, these were concatenated into a single input sequence.
2. *Key signature normalization*: All MIDI files were transposed so as to start on either C major or A minor, depending on the mode of the initial key. In cases where the MIDI file didn’t have a specified key signature, key estimation was performed.
3. *Inter-onset interval (IOI) quantization*: This consisted of approximating the ticks-per-quarter-note (TPQN) value of IOIs to the nearest value from a list of allowed durations — in this case, multiples of 16th-, 24th-, 20th-note subdivisions. This allowed the model to better generalize rhythmic information present in the input data.
   * 1. Feature extraction

Each Markov state encoded the following 4 note event features:

1. *Pitch*: the MIDI pitch value, in the range of 0-127.
2. *Forward note-wise IOI*[[3]](#footnote-3): temporal distance between the current and the next note. A zero value would indicate that the next note is synchronous with the current note — i.e., both belonging to the same chord.
3. *Backwards chord-wise IOI:* temporal distance between the current note, and the previous chord.
4. *Forward chord-wise IOI*: temporal distance between the current note, and the next chord.

The last two features were particularly useful in capturing contextual information that would otherwise be missed by only focusing on note-level features.

1. CONCLUSIONS AND FUTURE WORK

This paper introduced and described a feedback-based implementation of a Markov process, capable of generating polyphonic music sequences with high levels of novelty and coherence, relative to regular *nth*-order Markov models. The simplicity of Markov models relative to SOTA deep-learning architectures serves as a “toy model” or framework for exploring potentially transferable solutions to the problem of maintaining long-term dependencies in generative music.

Some possible avenues for improvement and experimentation include the use of other traditional reinforcement learning techniques, such as *noise injection* or *epsilon-greedy Q-learning*, which might further increase the novelty of the generated sequences by promoting exploration during generation. Another option to be tested is implementing this approach with Hidden-Markov Models to generate sequences with more complex musical features.

More broadly, while deep learning architectures dominate current research, it's worth considering the foundational and educational role that Markov models have played in the field of symbolic music generation, and how improving upon simpler and historically traditional algorithms might offer new insights that are applicable to the current state of the art.

1. REFERences

[1] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A Generative Model for Music.” arXiv, Apr. 30, 2020. doi: 10.48550/arXiv.2005.00341.

[2] Ö. Sandred, *The Musical Fundamentals of Computer Assisted Composition*, 2nd ed. Audiospective Media, 2022.

[3] A. Agostinelli *et al.*, “MusicLM: Generating Music From Text.” arXiv, Jan. 26, 2023. doi: 10.48550/arXiv.2301.11325.

[4] J. Copet *et al.*, “Simple and Controllable Music Generation.” arXiv, Jan. 29, 2024. doi: 10.48550/arXiv.2306.05284.

[5] S. Forsgren and H. Martiros, “Riffusion - Stable diffusion for real-time music generation,” 2022, [Online]. Available: https://riffusion.com/about

[6] P. Bodily and D. Ventura, “Steerable Music Generation which Satisfies Long-Range Dependency Constraints,” *Transactions of the International Society for Music Information Retrieval*, vol. 5, no. 1, pp. 71–86, Mar. 2022, doi: 10.5334/tismir.97.

[7] C.-Z. A. Huang *et al.*, “Music Transformer: Generating Music with Long-Term Structure,” presented at the International Conference on Learning Representations, Sep. 2018.

[8] L. Hiller and L. M. (Leonard M. Isaacson, *Experimental music; composition with an electronic computer*. New York, McGraw-Hill, 1959. Accessed: Apr. 01, 2023. [Online]. Available: http://archive.org/details/experimentalmusi00hill

[9] A. Sigit Ramanto and N. Ulfa Maulidevi, “Markov Chain Based Procedural Music Generator with User Chosen Mood Compatibility,” 2017, doi: 10.20668/adada.21.1\_19.

[10] A. Yanchenko, “Classical Music Composition Using Hidden Markov Models,” 2017.

[11] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep Unsupervised Learning using Nonequilibrium Thermodynamics.” arXiv, Nov. 18, 2015. doi: 10.48550/arXiv.1503.03585.

1. The source code and examples are available on <https://github.com/felipetovarhenao/markov-feedback/>. [↑](#footnote-ref-1)
2. <https://felipetovarhenao.github.io/markov-feedback/> [↑](#footnote-ref-2)
3. All inter-onset interval values were quantized and expressed in MIDI ticks-per-quarter-note units. [↑](#footnote-ref-3)