ATIAM 2018 - ML Project
Multi-step generation of chord progressions for inference in jazz through recurrent neural networks

Tristan Carsault, Jérôme Nika, Philippe Esling*
Institut de Recherche et Coordination Acoustique Musique (IRCAM)
UPMC - CNRS UMR 9912 - 1, Place Igor Stravinsky, F-75004 Paris
{carsault, jnika, esling}@ircam.fr

Abstract

This project aims to generate chord progressions of jazz music, represented as coherent chord label sequences with the help of probabilistic models. The motivation for this approach comes from a recent article on text-based LSTM networks for automatic music composition Choi et al. [2016]. In this paper, the authors use a recurrent neural network (RNN-LSTM) to generate symbolic chord sequences. They focus on two different approaches named *word-RNN* and *char-RNN*. These two variants use the same model architecture but rely on different learning methods. In this project, we will improve the word-RNN approach by doing multi-step prediction and by injecting music theory knowledge through the learning method in order to be able to perform accurate prediction of chord sequence and jazz melody generation. Ultimately, this project could be used to perform automatic accompaniment and improvisation.

1 Introduction

Symbolic music generation is a field that has been widely explored through Hidden Markov Models (HMM). For instance, Paiement, Bengio and Eck published a probabilistic models for melodic prediction using chord instantiation Paiement et al. [2005a]. Nevertheless, HMMs are bounded by their order of complexity. On the opposite, deep learning techniques are capable of computing highly non-linear functions and can extract information from complex data. Therefore, most recent works in the symbolic music generation field are now based on Neural Networks (NN), and particularly on Recurrent Neural Networks (RNN).

One of the largest issue in NNs is its incapacity to correctly store temporal information. However, in our setup, we need to discover the underlying structure of symbolic musics. Therefore, we need to design our model with an understanding of time. Recurrent Neural Networks (RNN) show promising results in many learning fields Graves [2013] including musical structure generation. To define recurrent networks, we usually rely on loops inside the network, where the output of a neuron is directly linked to itself for the next sequence element. The training of RNNs is performed by Backpropagation Through Time (BPTT). This algorithm works by fixing an amount of time steps, unrolling the RNN in order to obtain a classic NN, and training it with a standard back-propagation.

Nevertheless, standard RNNs tend to be unable to process long-term relationships correctly and quickly forget information across time steps. Hence, gated memory mechanisms solved this problem by adding trainable gates to standard RNNs that allow to select, stock and output the most important information for the task at end. The two most used gated RNN structures are the Long Short-Term Memory (LSTM) and the Gated Recurrent Unit (GRU).

*https://esling.github.io/atiam-ml-project

2 RNN in Pytorch

Your first approach with RNN will be done by using them on simple data to understand how they work, and to decide what framework to use for the future.

Exercise 1: Learn PyTorch and useful libraries. This exercise only includes learning basic programmation skills that you will need for the following work.

1. Install and learn PyTorch through basic tutorials `http://pytorch.org/tutorials/`
2. Create your name generator model with the help of this tutorial on RNN/LSTM `http://colah.github.io/posts/2015-08-Understanding-LSTMs/` and this Pytorch tutorial `https://pytorch.org/tutorials/intermediate/char_rnn_generation_tutorial.html`
3. Install and read about the support libraries for symbolic music anlysis : Music21[2] for general computer-aided musicological analysis and ACE analyzer [3] for musical relationships analysis between chords.

Exercice 2 - Understanding the reference paper After this stage, our model will be trained on the *real book dataset* Choi et al. [2016] composed of 2847 tracks taken from the real book [4] and by using the model described in the reference paper. The starting code in Keras and the dataset are available at this link : `https://github.com/keunwoochoi/lstm_real_book`.

You can also find the article `https://arxiv.org/pdf/1604.05358v1.pdf` and blog post related to this `https://keunwoochoi.wordpress.com/2016/02/19/lstm-realbook/`.

1. Based on the paper, try to re-implement in PyTorch the model (word-RNN) with RNN, LSTM and GRU.
2. Train and evaluate your models on the real book set.

3 Toy dataset

Exercice 3 - Designing a toy dataset In the field of symbolic music generation and chord inference, some typical constructions exist, while other remain under-studied. Therefore, defining a toy dataset is an open question and your proposed solution will be evaluated on the design and justification of your choices. Hence, we highly recommend that you use both your personal knowledge, music theory analysis books and eventually to propose some innovative approaches to the question.

In this case, the following questions must be answered separately for *chord progressions* and in terms of *harmonic function*

1. Describe the chord representations or harmonic descriptors that can be used by any system trying to tackle the question of musical inference
2. Describe the main functions of the Python library *Music21* that can be used to extract chord and harmonic information from midi
3. Describe situations in which these sequences require multiple scales of time
4. Code different *procedural functions* that could generate large sets of examples following all of your observations in the previous questions and by using the Lakh midi dataset[5]
5. Generate your sets of data in a structured way with the .xlab format (<bar>:<beat> <start-time> <endtime> <chord> <scaledegree> <key> <other harmonic descriptors ?>)
6. Explain how these sets can be analyzed and make statistics on them (number of similar chords, known harmonic scales)

[2] `http://web.mit.edu/music21/`

[3] `http://repmus.ircam.fr/dyci2/ace_analyzer`

[4] The real book dataset is a compilation of jazz classics that has been firstly collected by the students of the Berklee College of Music during the seventies. As of today we count a lot of existing books (e.g Realbook (1, 2, 3), New Realbook (1, 2, 3), the Fakebook, the real Latin book)

[5] `https://colinraffel.com/projects/lmd/`

```
<chord>      ::=    <pitchname> ":" <shorthand> ["("<ilist>")"]["/"<interval>]
                  | <pitchname> ":" "("<ilist>")" ["/"<interval>]
                  | <pitchname> ["/"<interval>]
                  | "N"

<pitchname>  ::=  <natural> | <pitchname> <modifier>

<natural>    ::=  "A" | "B" | "C" | "D" | "E" | "F" | "G"

<modifier>   ::=  "b" | "#"

<ilist>      ::=  ["*"] <interval> ["," <ilist>]

<interval>   ::=  <degree> | <modifier> <interval>

<degree>     ::=  <digit> | <digit> <degree> | <degree> "0"

<digit>      ::=  "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

<shorthand>  ::=   "maj" | "min" | "dim" | "aug" | "maj7" | "min7" | "7"
                  | "dim7" | "hdim7" | "minmaj7" | "maj6" | "min6" | "9"
                  | "maj9" | "min9" | "sus2" | "sus4"
```

Figure 1: Syntax of Chord Notation in the Realbook dataset, image taken from Harte [2010]

Exercice 4 - Reduce the chord vocabulary The original *word-RNN* described in the paper uses a vocabulary of 1259 labels. This great diversity comes from the precision level in the chosen syntax (Figure 1). Starting with the assumption that the generation of coherent chord progressions only needs information on harmonic functions (without taking harmonic enrichments into account), we propose first to apply a clustering of the elements from the initial alphabet A_0 to obtain three different alphabets. They represents different hierarchical organization corresponding to the harmonization of the major scale using triads or tetrachords, which is often used to write chord progressions (code for chord reduction in the ACE analyzer framework) :

- A_1 : Major, minor, diminished: N (which means no chord), maj, min, dim;
- A_2 : Major, minor, seventh, diminished: N, maj, min, maj7, min7, 7, dim, dim7;
- A_3 : Major, minor, seventh, diminished, augmented, suspended: N, maj, min, maj7, min7, 7, dim, aug, sus;

1. Train and evaluate your models with different types of reduction on the Realbook dataset and the toy dataset
2. What happens if you train the paper model on the real book set and then just fine tune it on the toy datasets ?
3. Bonus : propose, argument your choices and implement other kind of chord reductions

4 Models and expected work

In this second part, we focus on the coherence of the generated sequences. Indeed, chord sequences on which our model is trained are often composed of highly repetitive chords. Thus, the easiest solution for any network would be to always predict the next element as simply repeating the last chord. In order to minimize this effect, Choi et al. introduce a diversity parameter that penalizes redundancies in the generate sequence by applying a basic sampling on the duplicated elements in the generated sequence. Besides, we observe an error propagation for the multi-step prediction. Indeed, when we use the LSTM for prediction, we do not use the ground-truth sequence and we sample from the joint distribution over the sequence regarding its conditional distribution given the previously generated samples. Thus, the predicted sequences diverge from sequences seen during training and the system generates stronger errors at each step of the prediction.

4.1 Multi-step prediction

Exercice 6 - Professor Forcing In this paper Lamb et al. [2016], authors propose to use a new algorithm to train recurrent network. They observe that the training is based on solely ground truths

labels, while the inference has to reuse predicted (therefore, potentially wrong) elements. Hence, the error propagation has to be performed through multi-step outputs. They propose to use an adversarial approach to train the model with a second model that we call the discriminator. This second network learns to classify between generated examples and ground-truth sequence.

1. Implement this article in PyTorch
2. Evaluate the different models through their prediction scores on multiple scales
3. Use the given code to evaluate the model with musical distances

Exercice 7 - Improving the cost function (Bonus) In the original paper Choi et al. [2016], the cost function is a categorical cross-entropy. We want to upgrade this cost function with music theory consideration. Thus, we will modify it in order to train our model with other chord distances

1. Implement the distance between chords proposed in this article Paiement et al. [2005b] (Paragraph 2. Representation) (The code for the chord to vector transformation will be provided)
2. (Bonus) Another distance of your inspiration based on the music/perceptive/cognitive theory
3. Integrate this distance in the learning of our models and evaluate them after training.

References

Keunwoo Choi, George Fazekas, and Mark Sandler. Text-based lstm networks for automatic music composition. *arXiv preprint arXiv:1604.05358*, 2016.

Jean-François Paiement, Douglas Eck, and Samy Bengio. A probabilistic model for chord progressions. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR)*, number EPFL-CONF-83178, 2005a.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Christopher Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, 2010.

Alex M Lamb, Goyal Anirudh, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

Jean-François Paiement, Douglas Eck, Samy Bengio, and David Barber. A graphical model for chord progressions embedded in a psychoacoustic space. In *Proceedings of the 22nd international conference on Machine learning*, pages 641–648. ACM, 2005b.