ATIAM 2020 - ML Project
Jazz standard generation using variational learning

Tristan Carsault[1], **Axel Chemla–Romeu-Santos**[1], **Philippe Esling**[1*]
[1]Institut de Recherche et Coordination Acoustique Musique (IRCAM)
UPMC - CNRS UMR 9912 - 1, Place Igor Stravinsky, F-75004 Paris
{carsault, chemla, esling}@ircam.fr

Abstract

Several musical genres such as pop, jazz, or classical, are based on a tight underlying harmonic background that defines the structure of the piece. This underlying progression then defines the piece at a higher level of abstraction, that mostly varies in synchronicity with the pulse. It can therefore be represented by a "chord sequence", each chord representing the harmonic content of a beat. In this project, our aim is twofold. First, we want to generate chord progressions of jazz music, represented as sequences of chord labels, with the help of machine learning based generative models. Secondly, we want to generate varying sequences of notes based on the generated chord sequence. Thus, we propose to train a Variational Auto-Encoder (VAE) on existing chord sequences in order to extract a multi-dimensional space that will distribute a set of existing chord sequences, that can afterwards be used to generate an entire harmonic structure. We will resort to recurrent neural networks, a specific type of neural networks used for high-dimensional sequence modeling, in order to encode the dynamical structure of chord progressions. We will then propose various arpeggiating mechanisms in order to convert this structure into an existing sequence of notes, convertible as MIDI sequences and hence playable in most audio software.

1 Introduction

The development of machine learning based generative systems that occurred within the last decade provided a flourishing amount of different models, applicable to most domains thanks to the increased computational power of their underlying technical background. Among them, the *variational auto-encoder* (VAE) is a versatile model that has shown simultaneously great generalization properties and good reconstruction abilities, despite its light structure. VAEs are based on two processes : an *encoding* process, projecting input data to an abstract representation called the *latent space*, and a *decoding* process, giving back the data corresponding a given latent position. This latent space can thus be understood as a compressed representation of the learned database, whose reduced number of dimensions can then be navigated freely to generate data corresponding with the underlying structure of data.

The idea of this project is then to leverage variational auto-encoding to generate chord sequences, that can be converted in note sequences using an arpeggiator. The first task in hence to train a VAE on a selected database, providing an invertible representation of the learnt sequences. However, choosing a straightforward training for jazz standards is non-trivial for several reasons[2] We will then resort to variants of *recurrent neural networks*, a specific type of neural networks developed specifically to

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

[2]we can argue : variable length, interest of modeling transitions between block of chords, a dauntingly high number of input dimensions.

model dynamical series. The second task will then consist in sampling this latent space to generate novel chord sequences, used as an input to an arpeggiator, to produce an end-to-end system from the representation to a sequence of notes.

2 Variational Auto-Encoder

In this project we will use PyTorch, a versatile machine-learning Python framework that you will use to code your Variational Auto-Encoder (VAE).

Exercise 0 : First bibliography

1. Put your glasses and read this nice tutorial on variational auto-encoding of Blei et al. [2017], and then some articles grounding the topic Kingma and Welling [2013], Higgins et al. [2017]
2. If you feel it, read other papers (check the provided bibliography).

Exercise 1 : Install and learn Pytorch

This exercise only includes basic programming skills that you will need for the following work. If we already have enough expertise, you can skip it but I'd rather recommend you to have a look.

1. Install PyTorch on your device if it isn't already done.
2. Follow the basic tutorials and learn how it works `http://pytorch.org/tutorials/`.
3. Learn how to implement basic models, therefore use `torch.nn` library.

Exercice 2 : Code the VAE

You will know code your own VAE (finally). You can either look at the main article's implementation, or rely on this nice tutorial `https://wiseodd.github.io/techblog/2016/12/10/variational-autoencoder/` with nice mathematical explanations, programmed in Keras (otherwise it's too simple).

1. Based on the tutorial or another source you will find, develop your very own VAE.
2. Test your model on the MNIST dataset. As the output is binary, formulate what should be its loss function.
3. Compare your models to the VAE results from Kingma & Welling Kingma and Welling [2013].
4. Implement a warm-up. For example β varying from 0 to 1.

3 Chord sequence generator

Exercise 3 - Chord representations for machine learning. Our different models will be trained on the *real book dataset* (Choi et al. [2016]) composed of 2847 tracks taken from the real book [3]. The dataset is available at this link : `https://github.com/keunwoochoi/lstm_real_book`.

The original chord alphabet 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 hierarchical reduction of the elements from the full chord alphabet A_0 to obtain three different chord alphabets (A_1, A_2, A_3) of increasing precision. They represent different hierarchical organization corresponding to the harmonization of the major scale using triads or tetrachords, which is the most common way of writing chord progressions (the mapping will be provided) :

[3] 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)

```
<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]

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, dim7, aug, sus;

In the case of symbolic data, several transforms can be made to turn them understandable by the VAE. One of the most common way is to use Categorical or Multinomial distributions, that can be understood as generalizations of resp. Bernoulli and Binomial distributions for multi-class problems.

1. Read about Categorical and Multinomial distributions (Wikipedia is fine on this one...) and check out their PyTorch implementations (module `torch.distributions`).
2. Extract some information on the dataset (total number of chords for each chord class, chord repetition within chord sequences, chord transition matrix, ...) in order to choose the most relevant alphabets for the sequences.
3. Propose a way to model chords as Categorical or Multinomial distributions. Make some propositions and discuss them with us, so we can make sure that you don't go the wrong way!

Exercise 4 - Data loading and training Now that you get the perfect probabilistic representation for jazz chords, we will train our VAE on our chord sequences. As a first experiment we will train our VAE on short chord sequences, as training on single chords would be insignificant for our purpose, and training on full standard is for the moment impossible for the reasons exposed in the preliminary introduction. Furthermore, this slicing will provide the baseline for the full sequence generation that will be tackled in the next section.

1. What is at your opinion the relevant number of chords that should be present in the sequence?
2. Set a list of important parameters of your model (data, architecture, hyperparameters...) and forecast their influence. Make sure that these parameters can be easily set up by the user in the training script.
3. Read the tutorial [4] and and implement your custom chord sequence dataloader.
4. Adapt your VAE to generate sequence of categorical / multinomial outputs, and define the training procedure (stopping criterion, ...)
5. Don't forget to add saving/plotting functions to monitor your training (the bravest could use the wonderful `torch.utils.tensorboard` utilitary).

[4] `https://pytorch.org/tutorials/beginner/data_loading_tutorial.html`

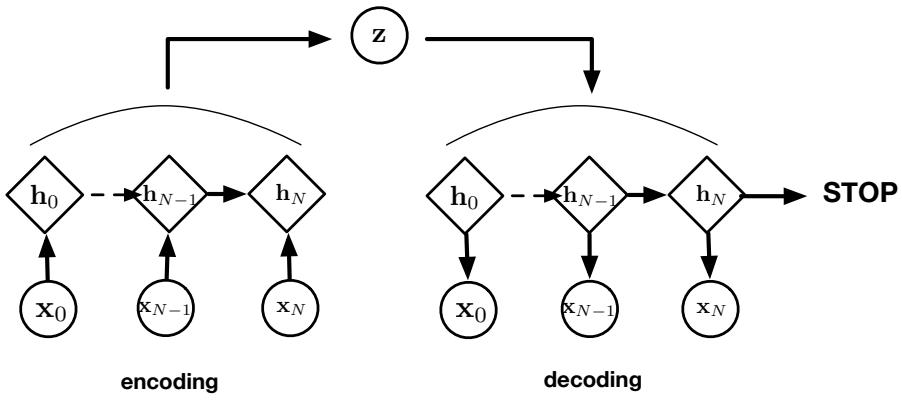

Figure 2: A recurrent variational auto-encoding needs a way to get a single latent vector from the output of a recurrent neural network. A "stop" criterion can also be given, in order to let the model decide the length of the generated sequence.

6. Verify your code works on a small subset of data, and send it to me so I can run them on our sweet GPUs. We will check the current state of your work at this point.
7. Propose evaluation procedures in order to check the performance of your VAE : reconstruction scores, latent space sampling, etc... to quickly check what your model can do.

4 Harmonic arpeggiator: generate notes from chord sequences

Exercise 5 - Arpeggiating. You can rely on the `music21` library to convert the output of your VAE to MIDI file. A code sample will be given, as this is not the core of your evaluation. But :

1. Select, randomly or using a pattern, a way to convert a given chord to a sequence of notes.
2. (optional but cool) make your sequence swing! add some ryhthmic flavour to your sequence.

5 Recurrent learning of chord sequences

Congratulations! You trained your first VAE on chord sequences. However, for the moment you are only able to generate short sequences from your latent space, which is not enough to generate a full standard. We will then find a way to "glue" several chord sequences into a single latent vector. This is possible with the *recurrent variational auto-encoder* formalism, that uses recurrent neural networks to encode a full sequence of data.

Exercise 6 - Recurrent neural networks. You will first have to get familiar with recurrent neural networks.

1. Read this nice tutorial on recurrent neural networks : `https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks`
2. If you can / are interested, check the supplementary material.
3. Discuss the difference between the different recurrent neural networks. Which one should be used?
4. Check their implementation in pytorch (package `torch.nn`), and propose quick test codes for RNN, GRU and LSTM modules.

Exercise 7 - Variational recurrent auto-encoding. Now, you will decide how to combine recurrent neural networks with variational auto-encoding. The idea here is to model a full score from a sequence of chord sequence slices. The problem is depicted fig. 5.

- Make some propositions to obtain a single latent space from the output of a recurrent neural network. Check this with us!
- What do you think of the corresponding Evidence Lower-Bound? Propose one, and justify it in the report.
- Find a way to encode / decode sequences of varying length, and discuss that in the report.
- Adapt your VAE to make it recurrent, and give us your code so we can train on our GPUs. As in exercise 4, provide all the training / evaluation routine, and show the generations / latent spaces in your report.
- Whatever they are, discuss the results. Which problems can you see here, and can you figure out their origins? How could them be fixed?

Advices

- As you count several members in your power team, do not hesitate to parallelize the work in order to save some time and to have things prepared when you need it. For example, exercise 3 and 5 can be made in parallel of exercises 2 and 4. You will move on much faster if you distribute and schedule these tasks among you!
- Make recurrent updates between all of you of your individual advances, such that everybody stay in touch and get ready to help another member on one task.
- The project is quite ambitious, but you will be evaluated more on your workflow and the developments / conclusion in the report than on the final results. We are here to help you, do not hesitate!
- **[Important]** we shall run your models on GPU, that can be then CUDA compatible. Do not forget to check the CUDA semantics here[5] in order to make your model CUDA compatible. Also, make your code as elegant as possible (using all the power of the `torch.nn` library), so that we are able to fix your code in case (a hellish style code means a hellish implementation score in the final grade..!).

References

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, Feb 2017. ISSN 1537-274X. doi: 10.1080/01621459.2017.1285773. URL `http://dx.doi.org/10.1080/01621459.2017.1285773`.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

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

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

[5] `https://pytorch.org/docs/stable/notes/cuda.html`