ATIAM 2018 - ML Project
Latent sequencing for dynamic musical patterns

Axel Chemla–Romeu-Santos[1,2] , Philippe Esling[1*]
[1] Università Degli Studie di Milano, Laboratorio d'Informatica Musicale (LIM)
[2] Institut de Recherche et Coordination Acoustique Musique (IRCAM)
UPMC - CNRS UMR 9912 - 1, Place Igor Stravinsky, F-75004 Paris
{chemla, esling}@ircam.fr

November 16, 2018

Abstract

Generative systems are machine-learning models whose training is based on two simultaneous optimization tasks. The first is to build a latent space, that provides a low-dimensional representation of the data, eventually subject to various regularizations and constraints. The second is the reconstruction of the original data through the sampling of this latent space. These systems are very promising because their space is a high-level, "over-compressed" representation that can be used as an intermediate space for several tasks, such as visualization, measurements, or classification. The main goal of this project is to use variational models in both audio and symbolic worlds, and to make them interact to have a end-to-end, full and controllable instrument.

1 Introduction

Among recent generative systems found in the literature, two have had a large success in the machine learning community. First, the *variational auto-encoder* (VAE) is based on a two-stage inference/generation procedure that showed great generalization properties and good reconstruction abilities despite of its light structure. VAEs are based on a twofold procedure : first an *encoding* pass, that projects input data to an abstract space called the *latent space*, and a *decoding* pass, that gives back the corresponding data from the latent position. Latent space can thus be understood as a *representation* for the learned database, that can be navigated freely to generate coherent data with the original data space.

We propose here to develop two different procedures : first, we will use this framework in the symbolic world by developing a rhythmic pattern generator,

*https://esling.github.io/

where we can generate stochastic patterns from the interaction with this latent space. Then, we will use it to generate the triggered sounds, in a way that this system is compatible such that variational pattern generator.

2 Variational Auto-encoder

First, you will have to learn PyTorch, a machine-learning Python framework that you will use to code your VAE. As you will use this code for both symbolic and signal generation, it has to be general enough to be adaptable in these two cases.

Exercise 0: Put your glasses on and read.

1. Learn how to read
2. Read the nice tutorial on variational auto-encoding of Blei [1], and the main articles of variational auto-encoder [2, 3, 4]
3. If you feel like it, read other papers in the project's bibliography

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

1. Install and learn PyTorch through basic tutorials `http://pytorch.org/tutorials/`
2. Learn how to implement basic models first "manually" (by registering parameters at hand), and then with the `torch.nn` library. Both are very important to code your model in a good way!

Please, don't make like your predecessors and code your model with the `torch.nn` module because it allows very easy GPU conversion, so will be able to send me your models to be trained on our delicious gpus. This will fasten your work a lot!

Exercise 2 : code the VAE. You will now code your very own variational auto-encoder. As the main article is quite cryptic on how to code this, you can rely on the tutorial `https://wiseodd.github.io/techblog/2016/12/10/variational-autoencoder/` that is in Keras but will give you a good intuition for the mathematics behind the system.

1. Based on the tutorial or another source you will find, develop your very own VAE

2. You have to code two different probability densities for the output distribution : Bernoulli (for binary output) and Gaussian (for continuous output), with their specific losses.

3. Compare your models to the VAE results from Kingma & Welling [4] on the MNIST dataset

4. Implement warm-up [6, 5] by yourself and make a qualitative analysis when varying the β parameter between 0 and 4.

5. Play with parameters, and make quick assumptions on the results

3 Pattern generator

Here, we first propose to train variational auto-encoders on *sequencer patterns* to extract a "rhythm" space, that can be navigated freely to interpolate between different patterns in real-time. First, this requires the construction of a dataset. Then, you will have to construct the Variational Bayes algorithm, and proceed to the training. Introduction of any label information can be involved during the training to provide a label-sensitive latent space.

Exercise 3: Make the sequencer dataset. The painful but mandatory part of machine learning : you will have to build your own dataset for pattern generation (as there is not any one on the internet). A fine way would be to catch some presets from VSTs, or some MIDIs somewhere, and to convert it into a bunch of matrices of size $n\_instruments \times n\_steps$ of 0. and 1. If the dataset you make come with some labels, that can be great!

I would advise you to do split your team between the ones building the control database and the ones building the sound database. Quicker it's done, better it is! See exercise 5

Exercice 4: Pattern generator training.

1. Use your VAE architecture to train it on your dataset. Here, the output has to be a Bernoulli distribution, as it the matrix is binary. The matrix has to be flattened to enter the MLP, and de-flattened at the output of the VAE!

2. Try with different latent spaces dimensionalities. Here the model is very light, as here we are working on Bernoulli distributions of little size. You can thus try with very few dimensions (between 2 and 8)

3. If you have some label information on the pattern, you find ways to condition the learning process on it.

4. Don't forget the evaluation part of this: train / test loss curves, latent space plotting, and reconstruction plots. Make it sweet to use!

4 Sound generation

Here we learn the sounds that will be triggered by the latent rhythm generator. Similarly, you will have to construct a dataset (can be different from drums). The training will be quite different that the pattern generator, as the dimensionality and the nature of the sounds implies much bigger networks. For that, you will have to use *convolutional* encoders / decoders that I will give you to perform the learning, and we will probably run it on our marvelous IRCAM GPUs. Another import detail is that you will have several categories of sound, and that the generation process must be controllable in this way.

Exercise 5: Make the audio dataset. Same pain than exercise 3 ; if you followed my advice, this have been already done!

1. Gather your audio files, labelled by type of sounds corresponding the rows of your sequencer.
2. We will use *Non-Stationary Gabor Transform*, with the handsome library of sir Grill[1]. This is better than STFT because it allows a custom scale (such that the CQT one), has much less dimensionality and is invertible.
3. As we will use convolutional encoders/decoders, the input data is not anymore 1D but 2D. You thus have to fix a grid size, and decide whether your signals have to be concatenated or given entirely to the VAE depending of their nature.

Exercise 6: Sound generation training.

1. I will give you the convolutional modules, that will be easily implemented in your model if you did it right. This time, the output has to be Gaussian.
2. Find the right conditioning, so we can target a specific class during generation to integrate it to the sequencer.
3. Verify your code works, and send it to me so I can run them on our sweet GPUs.

5 Finally some fun

Exercise 7: Make it work!

[1] `https://github.com/grrrr/nsgt`; this is unfortunately for Python2, i will give you the weakly corrected version of mine for Python3

1. Plug your two VAEs together, so you can generate full loops by choosing a point in your sequencer's latent space!

2. Generate a lot of items, trying to define a proper exploration strategy

Exercise 8: Make it fun!

1. We now have a latent sequencer with a variational synthesizer. Now that the main core is developed, how could you improve the interaction of a user with this synthesizer (additional knobs, additional options, randomization, you are free about almost anything...!) ?

2. Make an article..?

References

[1] *Variational inference: A review for statisticians*, Blei, David M and Kucukelbir, Alp and McAuliffe, Jon D, Journal of the American Statistical Association, volume 112 issue 518, 2017

[2] *Extracting and composing robust features with denoising autoencoders*, Vincent, Pascal and Larochelle, Hugo and Bengio, Yoshua and Manzagol, Pierre-Antoine, Proceedings of the 25th international conference on Machine learning, 2008

[3] *Stochastic backpropagation and approximate inference in deep generative models*, Rezende, Danilo Jimenez and Mohamed, Shakir and Wierstra, Daan,arXiv:1401.4082,2014

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

[5] *beta-vae: Learning basic visual concepts with a constrained variational framework*, Higgins, Irina and Matthey, Loic and Pal, Arka and Burgess, Christopher and Glorot, Xavier and Botvinick, Matthew and Mohamed, Shakir and Lerchner, Alexander, 2016

[6] *How to train deep variational autoencoders and probabilistic ladder networks*, Sønderby, Casper Kaae and Raiko, Tapani and Maaløe, Lars and Sønderby, Søren Kaae and Winther, Ole, arXiv:1602.02282, 2016