ATIAM 2017 - ML Project
Latent representations for real-time synthesis space exploration

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

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 develop variational models to find generative *sound synthesis space*, where each point of this space correspond to a new data content that comes from the *high-level* understanding of the input data.

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 Kingma and Welling [2013]. The second is the *adversarial generative system* (see Goodfellow et al. [2014]), that showed impressive reconstruction abilities but rather poor latent expressivity.

This project aims to study the use of variational auto-encoders to learn a significant latent space for sound synthesis, and to build a real-time synthesizer that can be used for artistic purposes. This study involves two main objectives

1. *Obtaining the highest quality of reconstruction*. This implies to test and evaluate the efficiency of learning on different sound representations (spectrograms, modulation spectrum, raw data, spectro-temporal receptive fields), and choose the most convincing one in terms of efficiency and sound quality.

2. *Obtaining the best latent representation in terms of control and expressiveness*. This means targeting the behavior of variational learning to develop heuristics on the latent space construction and to study various regularizations constraints.

*https://esling.github.io/

2 Toy dataset

Your first approach with variational auto-encoders will be done by using them on simple data to understand how they work, and to decide what framework to use for the future. This work will be split in three important tasks.

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. Install and read about the support libraries for DSP processing: LibRosa[2] for analysis and Pyo[3] for modular sound generation

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 VAE.
2. Compare your models to the VAE results from Kingma & Welling Kingma and Welling [2013] on the MNIST dataset
3. Implement warm-up Sønderby et al. [2016] by yourself and make a qualitative analysis when varying the β parameter between 0 and 4.
4. Play with parameters, and make quick assumptions on the results

Exercise 3: Develop a toy modular synthesis set for VAEs. To understand well the behavior of the VAE with audio data, the first step will be to procedurally build toy datasets to feed VAE algorithms with simple architectures. This step aims to catch how it behaves to simple auditory factors of variation. Hence, you must first define all the simple factors of variations that may exist in synthesis sounds. Then, the examples must be conceived to test independently each factor of variations, and then to test gradually combinations of both. For example, in image world it would be translation, rotation, shape, etc. Reversely, it will be asked to test several VAE architectures on the same toy dataset, to analyze the coincidence between the VAE structure and the handling on a particular factor. A very interesting example of toy dataset for image that you can draw inspiration from is the *dSprites* dataset `https://github.com/deepmind/dsprites-dataset`. You should rely on the *Pyo* library to generate increasingly complex sounds.

1. Propose all factors of variation in audio that you find relevant for sound synthesis.
2. Build procedurally as much toy datasets as you think is necessary to represent clearly the related factor of variation
3. Test it in your VAE, and make a qualitative & quantitative analysis of the results
4. Test combinations of these factors in a way you will justify, and draw some conclusions compared to the "mono-factor" results.
5. Find interesting visualizations that could exhibit this behavior
6. (optional) what if we do the opposite? Take a VAE trained on full data, and fill it with your toy examples. Is the corresponding factor of variation well preserved?

3 Models and expected work

We will now aim to design a real-time sound synthesis system based on the variational auto-encoder framework. The system will be conceived on a Python - Max bridge, communicating by OSC protocol, whose code will be provided. The next step will be the in-depth study of the generative process to ameliorate the model in terms of interaction and expressiveness. At the end, you will

[2] `https://librosa.github.io/librosa/`

[3] `https://github.com/belangeo/pyo`

also have to decide on which final dataset you will apply to your system, depending the preliminary studies from exercise 1 and the application you want for your synthesizer.

Exercise 4: Test various sound representations. Another important part of this first-time procedure is the use of various sound representations. Indeed, contrary to images where we directly give raw information to the model, we usually rely on specific representations of audio signals, except for raw data. Furthermore, in our case these representations must be *invertible* so we can recover an audible sound signal. Several sound representations will be thus tested to choose which ones are convincing for real-time synthesis, in terms of sound quality and learning performance.

1. Draw a list of possible sound representations that we could give to the VAE. Remember that we have to convert output in an audible sound signal. We will mainly focus on spectrograms, modulation spectrums and spectro-temporal responsive fields, except if you have other ideas
2. Propose a stochastic probability distribution that could model this representation.
3. Find a code to compute them (with LibRosa, another library, or your own code)
4. Launch tests of various representations with the same VAE architecture, and evaluate them.

Exercise 5 : latent space regularizations. As sound synthesis is achieved by the exploration of the VAE latent space, its morphology is thus very important in terms of expressiveness and interaction. We will thus go deeper into the understanding of the latent space construction by exploring the following items :

regularization strategies : we can influence the variational learning of the latent space by adding specific regularization terms and constraints, that will directly change its morphology.

1. try to replace the Kullback-Leibler divergence in the ELBO term by other divergences (α-divergences, Jensen-Shannon divergence, maximum mean discrepancy, if you have another idea...!)
2. test and evaluate the results for your toy dataset in terms of results and sound results, and conclude.
3. try to train VAE with high latent dimensions on your toy dataset. Analyzing the distributions defined by all of your examples outputted by the encoder, can you propose a way to determine the number of dimensions required by the vanilla VAE to correctly encode the inherent factors of variation?
4. (bonus) propose a mathematical justification of the obtained effects when replacing the KLD by another divergence, by reformulating the lower-bound expression.

Exercise 6 : dimensionality reduction. The VAE regularizes better data information when the number of latent dimensions if more important ; however this can be impraticable with user control. We can fix this by selecting the most relevant dimensions by variance analysis, or applying manifold analysis such as PCA (see Tipping and Bishop [1999]), Laplacian eigenmaps (see Levin and Lyzinski [2017]), Locally Linear Embeddings (see Roweis and Saul [2000], Saul and Roweis [2000]) to get the most relevant 2D-embedding from an higher-dimensioned space.

1. train a VAE with a high number of latent dimensions (32, or 64 for example) and, if not done in the previous section, analyze the distributions defined by all of your examples outputted by the encoder and evaluate the number of dimensions required by the vanilla VAE to correctly encode the inherent factors of variation.
2. choose a manifold learning technique among the one presented above, and read the corresponding article
3. implement it and test it on virtual toy dataset (no correspondance with image nor sound is required, it is just to check if your algorithm is working)
4. apply it on the latent space of your VAE, where the points are the projections given by the encoder (take the mode of the distribution) to obtain a low-dimension manifold. Try with several manifold dimensions.
5. implement it in the real-time environment, and evaluate the control space given by the manifold.

bonus exercise : generate raw audio! If there is a little time left, we will now try to generate raw audio, to see how the VAE handles this.

1. study Bowman et al. [2015], and code your own autoregressive decoder.
2. can you find a way to transform your continuous output in a classification problem? Formulate it.
3. try it with very small portions of raw audio (let's say, 1024 samples max).
4. analyze the obtained latent spaces and reconstruction results, as you did in the previous sections.

References

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

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*, pages 2672–2680, 2014.

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

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

Keith Levin and Vince Lyzinski. Laplacian eigenmaps from sparse, noisy similarity measurements. *IEEE Transactions on Signal Processing*, 65(8):1988–2003, 2017.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

Lawrence K Saul and Sam T Roweis. An introduction to locally linear embedding. *unpublished. Available at: http://www. cs. toronto. edu/˜ roweis/lle/publications. html*, 2000.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.