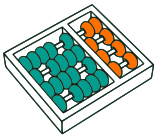

UNICAMP

Flow Models

Lesson No. 3

Levy Gurgel Chaves [*]

1 Introduction

Generative models are a hot and very popular topic in the machine learning literature. Flows methods are one of subclass of generative models which can be used to learn and model probability distributions.

Flows based methods permit, for example, take a sample from a simple distribution – Gaussian, for example – and then transform those samples using a bijective function which performs a change on variable in probability distribution. Doing this process many times can result in a complex PDF for the transformed random variable. This and some mathematical proof and ideas from normalizing flows can be seen in [1].

The main idea behind is to express $x \in \mathbb{R}^d$ a continuous, real and random variable as a transformation f of a real vector z sampled from a simple distribution p_z. Symbolically, we have $x = f(z)$ with $z \sim p_z$

2 Change of Variable Theorem

When we say about the tractability of flow models, we say that each function f applied has to be invertible (bijective) and differentiable.

Given a random variable z and its known PDF and $f : \mathbb{R}^d \to \mathbb{R}^d$ a invertible function, we can transform a probability distribution using a composition of k invertible functions. If $z \sim p(z)$ the resulting random variable $x = f(z)$ has the following probability distribution [1]:

$$p_x(x) = p_z(z) \left| det J_f(z) \right|^{-1} \tag{1}$$

where $J_f(z)$ is the Jacobian of f at z, that is a $j \times j$ matrix of partial derivatives.

Thus, a series of functions $f_k, k \in 1, 2, ..., k$ can be applied and and obtain a normalized flow as bellow:

$$\mathbf{x} = \mathbf{z}_k = f_k * f_{k-1} * \ldots * f_1(z) \tag{2}$$

$$z_k \sim p_k(z_k) = P_0(z_0) \prod_{i=1}^{k} \left| det \frac{\partial f_i(z_{i-1})}{\partial z_{i-1}} \right|^{-1} \tag{3}$$

Through this series of transformations we are able to transform a simple probability distribution into a complicated one. In order to train our model, the Jacobian is expected to be easy to compute, not costly and tractable. So every transformation has to be designed to meet these requirements.

With normalizing flows, the exact log-likelihood of input data becomes tractable. As a result, the training of flows models is simple the negative log-likelihood over the training dataset.

3 Autoregressive Flows

Autoregressive models try to model a sequential data , where each ouput depends on the past data but not in the future ones. In resume, the probability of observing X_i is conditioned on $X_1, X_2, ..., X_{i-1}$ and the final conditional density probability is the product of these conditional probability.

[*]RA: 264958 - l264958@dac.unicamp.br

In order to achieve this balance, neural autoregressive flows decomposes autoregressive flows into two components: an autoregressive conditioner, and an invertible transformer. The transformer (pictured below in red) can be any invertible function. The conditioner (in blue) is another neural network that outputs parameters for the transformer at each step, as a function of the preceding input features (X_i). In principle, any neural architecture that satisfies the autoregressive property can be used as the conditioner. Famous autoregressive flows are MADE [2], PixelCNN [3] and WaveNet [4].

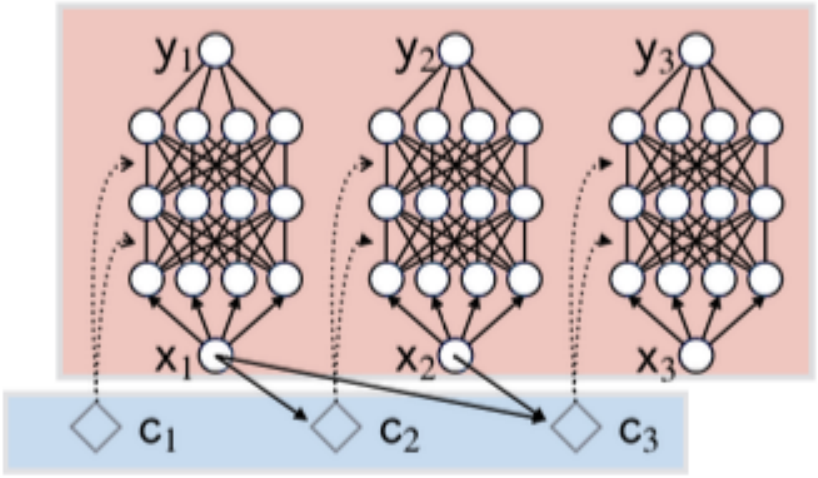

Figure 1. neural autoregressive flows

4 Non-autoregressive Flows

Generally, methods that do not use the autoregressive tends to use the idea of Coupling Layers. Coupling methods aimm to produce expressive transformations for flows. Consider a disjoint partition of the input $\mathbf{x} \in \mathbb{R}^D$ into two subspaces: $(\mathbf{x}^a, \mathbf{x}^b)$ and a bijection function $f(\cdot, \theta)$ parametrized by θ. Then, the partition is:

$$\mathbf{y}^a = f(\mathbf{x}^a, h(\mathbf{x}^b)) \tag{4}$$

$$\mathbf{y}^b = \mathbf{x}^b \tag{5}$$

where θ is defined by any function which only takes $\mathbf{x}^b$ as input. The bijection f^{-1} is called *coupling function* and the resulting function g after the application of some functions. Logically, we choose an f that is bijective, then:

$$\mathbf{x}^a = f^{-1}(\mathbf{y}^a, h(\mathbf{x}^b)) \tag{6}$$

$$\mathbf{x}^b = \mathbf{y}^b \tag{7}$$

The Jacobian each transformation f^{-1} is given by the a triangular matrix where the determinant is simply the diagonal of that matrix. Some papers that follow this idea are Real-NVP [5] and Flow++[6]. For more details about the proof the reader can check [7].

5 Conclusion

Flow Models are a powerful methods among those in generative models. The method has proved to be a good generative model and the result of some methods can be compared with the GAN methods. The field has recently emerged and still has several open points. Such as, scalability for large images, videos and multiple modalities, creation of new inverse functions.

References

[1] D. J. Rezende and S. Mohamed, "Variational inference with normalizing flows," *arXiv preprint arXiv:1505.05770*, 2015.

[2] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," in *Advances in Neural Information Processing Systems*, 2017, pp. 2338–2347.

[3] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, "Conditional image generation with pixelcnn decoders," in *Advances in neural information processing systems*, 2016, pp. 4790–4798.

[4] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[5] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *arXiv preprint arXiv:1605.08803*, 2016.

[6] J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel, "Flow++: Improving flow-based generative models with variational dequantization and architecture design," *arXiv preprint arXiv:1902.00275*, 2019.

[7] L. Weng, "Flow-based deep generative models," *lilianweng.github.io/lil-log*, 2018. [Online]. Available: http://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html.