An introduction of Autoregressive flows

Recently, I am trying to apply normalizing flows to the time-series problems, and I found that Autoregressive flows are more suitable for time-series modelling than coupling flows.

The joint distribution $p(\boldsymbol{x})$ of a known time series can be expressed as follows:

$$p(\boldsymbol{x}_{1:T}) = p(\boldsymbol{x}_1)p(\boldsymbol{x}_2|\boldsymbol{x}_1)p(\boldsymbol{x}_3|\boldsymbol{x}_2,\boldsymbol{x}_1)p(\boldsymbol{x}_4|\boldsymbol{x}_3,\boldsymbol{x}_2,\boldsymbol{x}_1)\ldots = \prod_{t=1}^{T} p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1})$$

This is an **auto-regressive model**, and each node depends on its predecessors in the ordering. Specifically, $\boldsymbol{x}_t$ is only dependent on the previous time $\boldsymbol{x}_{1:t-1}$.

It is suitable to use **autoregressive flows** to transform this joint distribution. And the dependency of special x_i can be expressed as:

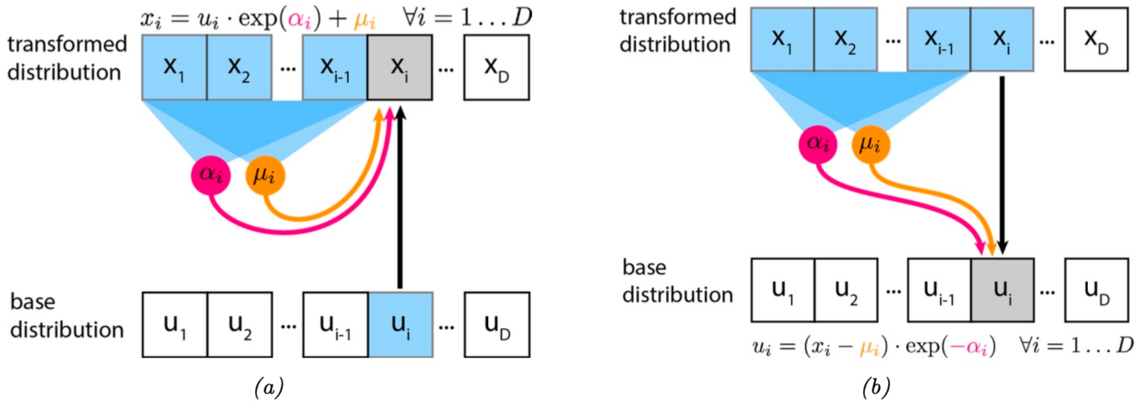
$$x_t = h(u_t; \Theta(\boldsymbol{x}_{1:t-1})), \qquad t = 1, \ldots, T.$$

where $\boldsymbol{u}$ contains T scalar elements, that is, $\boldsymbol{u} = (u_1, \ldots, u_T) \in \mathbb{R}^T$.

Specifically, we assume h is affine transformation, and Θ is a designed neural network. Its input is $\boldsymbol{x}_{1:t-1}$, and it outputs two scalar values α_i and μ_i, and they are used as scale and offset parameters in the affine transformation:

$$x_i = u_i \exp(\alpha_i) + \mu_i$$

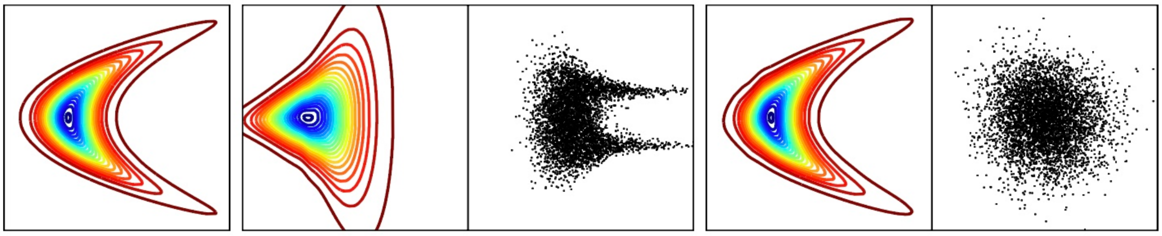
Affine autoregressive flows is illustrated in the following figure:


$x_i = u_i \cdot \exp(\alpha_i) + \mu_i \quad \forall i = 1 \ldots D$
transformed distribution
x_1
x_2
...
x_{i-1}
x_i
...
x_D
α_i
μ_i
base distribution
u_1
u_2
...
u_{i-1}
u_i
...
u_D
$u_i = (x_i - \mu_i) \cdot \exp(-\alpha_i) \quad \forall i = 1 \ldots D$

(a) (b)

Figure 24.4: (a) Affine autoregressive flow with one layer. In this figure, $\boldsymbol{u}$ is the input to the flow (sample from the base distribution) and $\boldsymbol{x}$ is its output (sample from the transformed distribution). (b) Inverse of the above. From [Jan18]. Used with kind permission of Eric Jang.

The order of dependency is important for autoregressive flows. Here is an example:

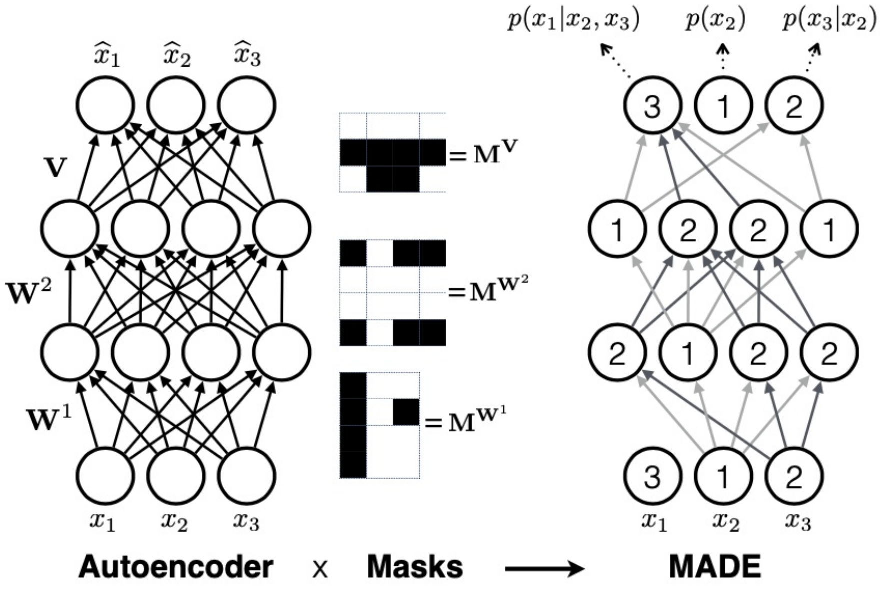


(a) Target density (b) MADE with Gaussian conditionals (c) MAF with 5 layers

Figure 1: **(a)** The density to be learnt, defined as $p(x_1, x_2) = \mathcal{N}(x_2 \mid 0, 4)\mathcal{N}(x_1 \mid \frac{1}{4}x_2^2, 1)$. **(b)** The density learnt by a MADE with order (x_1, x_2) and Gaussian conditionals. Scatter plot shows the train data transformed into random numbers $\mathbf{u}$; the non-Gaussian distribution indicates that the model is a poor fit. **(c)** Learnt density and transformed train data of a 5 layer MAF with the same order (x_1, x_2).

In the figure, (a) is the target density, where x_1 is depends on x_2. In (b), we model the dependency with inverse order, resulting in poor results. However, in (c), we can use a deep model to avoid this poor fitting.

We have already discussed that Θ is a designed neural network, and its input is $\boldsymbol{x}_{1:t-1}$. Naively, we need to train a neural network for each $\boldsymbol{x}_{1:t-1}$ for $t \in T$. Germain et al. proposed using a designed mask to improve efficiency. In his method, it will use only one neural network, illustrated as:


$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$
$\mathbf{V}$
$\mathbf{W}^2$
$\mathbf{W}^1$
x_1 x_2 x_3
$= \mathbf{M}^{\mathbf{V}}$
$= \mathbf{M}^{\mathbf{W}^2}$
$= \mathbf{M}^{\mathbf{W}^1}$
$p(x_1|x_2, x_3)$ $p(x_2)$ $p(x_3|x_2)$
3 1 2
1 2 2 1
2 1 2 2
3 1 2
x_1 x_2 x_3
Autoencoder x Masks → MADE

The right part of the figure is the modified neural network. The number in the circle node is the dependency order (this is hyperparameters, and can be assign to different order in each layer of flows), which represent the dependency: (x_2, x_3, x_1), that is, x_1 depends on x_2, x_3, and x_3 depends on x_2. The relationship is also reflected in the inner connections between neurons in the neural network.

Here, we use the idea of the mask to modify the connection within the neural network but change the output of conditional probability to affine parameters: $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$

The following code is a straightforward implementation of Autoregressive flows. I will adapt this code and use it in our project:

https://github.com/ikostrikov/pytorch-flows/blob/master/flows.py (https://github.com/ikostrikov/pytorch-flows/blob/master/flows.py)

References

Germain, M., Gregor, K., Murray, I. & Larochelle, H. (2015). MADE: Masked Autoencoder for Distribution Estimation. *Proceedings of the 32nd International Conference on Machine Learning*, pages 881–889.

Jang, E. (2018) Normalizing Flows Tutorial. Retrieved from https://blog.evjang.com/2018/01/nf2.html

Murphy, K.P. (in press). *Probabilistic Machine Learning: Advanced Topics*. MIT Press.

Papamakarios, G., Pavlakou, T. & Murray, I. (2017). " Masked Autoregressive Flow for Density Estimation". In: *NIPS*.