Deep Probabilistic Generative Models - Real NVP

M2 ANO BEHIDJ Ramzi
M2 AI JEMLI Nour

October 2021

Contents

1 Introduction

Normalizing flow is a kind of Implicit model given latent probability distribution $z \sim P(z)$. Instead of defning explicitly the resulting distribution of x, we use an invertible mapping function $g(z;\theta)$ to transform the distribution of the latent space to the target sample space. In normalizing flows, we wish to map simple distributions (easy to sample and evaluate densities) to complex ones (learned via data). The change of variables formula describe how to evaluate densities of a random variable that is a deterministic transformation from another variable. Change of Variables: Z and X be random variables which are related by a mapping $g : \mathbb{R}^n \to \mathbb{R}^n$ such that $X = g(Z)$ and $Z = g^{-1}(X)$. Then

$$p_X(x,\theta) = p_Z\left(g^{-1}(x,\theta)\right)\left|\det\left(\frac{\partial g^{-1}(x,\theta)}{\partial x}\right)\right|$$

The name "normalizing flow" can be interpreted as the following:

1. "Normalizing" means that the change of variables gives a normalized density after applying an invertible transformation.
2. "Flow" means that the invertible transformations can be composed with each other to create more complex invertible transformations.

More generally, in multiple dimensional cases, the derivative of g^{-1} with respect to x is the determinant of Jacobian matrix J_x of $g^{-1}(x,\theta)$. Thus we can rewrite the equation as:

$$p(x;\theta) = p(g^{-1}(x,\theta))\left|\det J_x g^{-1}(x,\theta)\right|$$

Finally we have the loss function of the normalizing flow:

$$\mathcal{L} = -\log(p(g^{-1}(x,\theta))) - \log\left(\left|\det J_x g^{-1}(x,\theta)\right|\right)$$

2 What problem does normalizing flows solve?

The normalizing fow model transforms the probability distribution of the latent space to the target sample space by directly performing an invertible mapping function $g(z;\theta)$; and it overcomes the intractability of computing likelihood based on assumptions of distribution.
To ensure the generator $g(z;\theta)$ to be invertible, it fixes its input z and the output x to have the same dimension. On top of that, normalizing flow introduces the coupling layer, which splits the generator function's input into two parts. And it transforms only one part of the inputs while copying the other because this will make the Jacobian Matrix a triangular matrix and, therefore, easy to compute its determinant.
The constraints on generator $g(x;\theta)$ (same dimension for input and output, coupling tricks) makes it possible to compute in practice, but also limit its expression. So we add a sequence of invertible mapping functions $g^{(i)}\left(\cdot \mid \theta^{(i)}\right)$ to improve the capability of generator g, where g is defined as:

$$g(z;\theta) = g^{(k)}\left(g^{(k-1)}\left(\ldots\left(g^{(1)}(z)\right)\right)\right)$$

3 Benefit of normalizing flow compared to other models

Compared to the VAE and SBN model that we've learned from the course, the normalizing flow is a kind of implicit generative model. It means that we need neither to make assumptions about $p(x;\theta)$, to deal with the intractability problem (computing the summation of continuous latent variable etc.)

The training loss of the normalizing flow model is relatively easier to compute, and we don't need any Monte Carlo sampling trick to approximate the distributions, which are expensive in computation and have dependency problem. The only thing required is to design the structure of mapping functions $g(z;\theta)$, which makes the determinants of the Jacobian matrix easy to compute.

4 What is the intuition behind NICE and Real NVP models?

For NICE (Non-Linear independant Components Estimation), instead of mapping all the latent variables to the target sampling space with the same function $g(z;\theta)$, we copy the first m variables of z, then shifts all the remaining ones with function $h(z;\theta)$:

$$\begin{aligned} x_{1:m} &= z_{1:m} \\ x_{m+1:n} &= z_{m+1:n} + h\left(z_{1:m};\theta\right) \end{aligned}$$

The Real NVP (Real-valued Non-Volume Preserving) is a variant based on the NICE model. We add a scaling function $s(z;\phi)$ on its exponential form for the copying part of the variables: (the shifting function here is written as $t(z;\theta)$)

$$\begin{aligned} x_{1:m} &= z_{1:m} \\ x_{m+1:n} &= z_{m+1:n} * \exp\left(s\left(z_{1:m};\phi\right)\right) + t\left(z_{1:m};\theta\right) \end{aligned}$$

The intuition behind all flow-based models is that we combine a sequence of invertible transformations to construct a generative model $g(z;\theta)$ to approximate the real data distribution p(x). Each mapping function is supposed to add non-linearity by dividing the inputs into two parts, and we only transform one part of them each time. So combining multiple layers can realize more complex non-linearity. More specifically, for NICE, we only do a translation transformation, so its volume does not change (log of the determinant of Jacobian always equals 1) unless we add a scaling layer. However, for Real NVP, we add a scaling function in an exponential form, with a translation transformation as seen in NICE to achieve an affine transformation. This transformation is kind of "non-volume preserving," and it improves its ability compared to NICE flows.

5 Results

The following figures represents the results obtained by Real NVP. The latent probability distribution was set to be two independent Normal distributions.

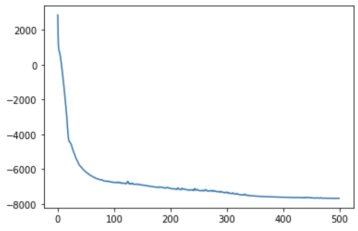

Figure 1: loss function

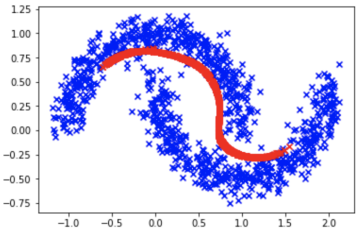

Figure 2: sample distribution

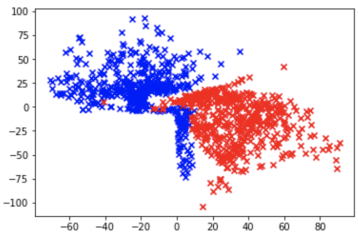

Figure 3: latent visualization

From the latent space, we can observe that the variables with different classes were very well classified, but the sampling results isn't good, After carefully verifying the results, we found that

the scale of the latent space was not what we expected, and the variance was much more significant than the Normal distribution. Then we found out that in the loss function, the absolute value of the determinant term was also much greater than the first log prob(z) term, which finally led to this "distortion" of the latent distribution.