Lab3 - Real NVP

Jiangnan Huang, You Zuo

November 13, 2020

Instructor: Caio Corro

1 Normalizing flow

Normalizing flow is a kind of **Implicit model** given latent probability distribution $z \sim P(z)$. Instead of defining 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.

By applying **change of variable theorem**, the proability distribution of x can then be written as:

$$P(x;\theta) = P(F(x;\theta))|\frac{d}{dx}F(x;\theta)| \tag{1}$$

where $F(x;\theta)$ is the invert function of $g(z;\theta)$.

More generally, in multiple dimensional cases, the derivative of F with respect to x is the determinant of **Jacobian** matrix J_x of $F(x;\theta)$. Thus we can rewrite the equation as:

$$P(x;\theta) = P(F(x;\theta))|detJ_xF(x;\theta)| \tag{2}$$

Finally we have the loss function of the normalizing flow:

$$\mathcal{L} = -log(P(F(x;\theta))) - log(|detJ_xF(x;\theta)|) \tag{3}$$

Q1.1 What problem does normalizing flows solve?

Answer:

1. The normalizing flow 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.

2. 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.

3. 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 $g^{(i)}(\cdot|\theta^{(i)})$ to improve the capability of generator g, where g is defined as:

   $$g(z;\theta) = g^{(k)}(g^{(k-1)}(\dots(g^{(1)}(z)))) \tag{4}$$

   And that is why it is called a "flow" because it's a sequence of mapping functions.

Q1.2 What is their benefit compared to other model we saw in the course?

Answer: 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 $p(x;\theta$, nor 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.

2 NICE

For NICE (No-Linear 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 $h(z;\theta)$:

$$\begin{aligned} x_{1:m} &= z_{1:m} \\ x_{m+1:n} &= z_{m+1:n} + h(z_{1:m};\theta) \end{aligned} \tag{5}$$

3 Real NVP

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 $t(z;\theta)$)

$$
\begin{aligned}
x_{1:m} &= z_{1:m} \\
x_{m+1:n} &= z_{m+1:n} * exp(s(z_{1:m};\phi)) + t(z_{1:m};\theta)
\end{aligned}
\tag{6}
$$

Q2 What is the intuition behind NICE and Real NVP models?

Answer: 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.

4 Results

The following figure 1 represents the results obtained by Real NVP. The latent probability distribution was set to be two independent Normal distributions.

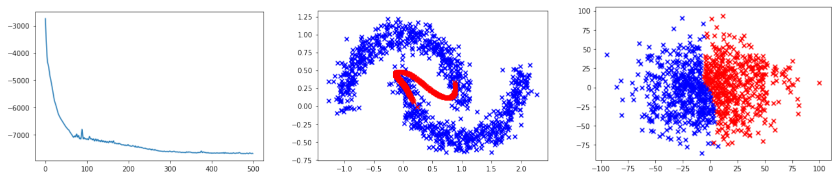

Figure 1: loss / samples / latent space

From the latent space, we can observe that the variables with different classes were very well classified, but the sampling results were not good, why? 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.

After several experiments, we finally manually added a trade-off parameter $\alpha = 0.001$ to the determinant term to balance the loss function, then we got some relatively better results.

Figure 2 shows the results obtained by manually adding a trade-off parameter $\alpha = 0.001$ to the determinant term of the loss function.

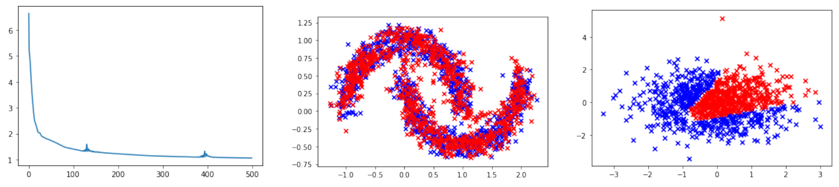

Figure 2: loss / samples / latent space after adding trade off param

But here comes a new question: as we already have a scaling function $s(z;\phi)$ in the Real NVP model, why our model can not fit the right scale of the latent distribution during training loops? We haven't found the answer for now.