Flow Based Deep Generative Models

Lecture Notes written by Chieh Wu

Tuesday, November 16, 2021

Introduction

Given some data X, Normalizing flow model is a powerful way to take the data X and identify the associated distribution $p(x)$ using neural networks.

The idea starts with two distributions

1. an Isotropic Multivariate Gaussian distribution $p(z)$.
2. our data distribution $p(x)$.

We make a crucial assumption that z and x are related by an invertible function f:

$$x = f(z), \quad z = f^{-1}(x),$$

where $f : Z \to X$ is a diffeomorphism (a smooth, invertible function with a smooth inverse). It has been discovered that if we know $p(z)$, there exists a direction relationship to $p(x)$ using the change of variables formula:

$$p(x) = p\left(z\right) \left| \frac{df^{-1}(x)}{dx} \right| \tag{1}$$

or equivalently

$$p(x) = p\left(z = f^{-1}(x)\right) \left| \frac{df^{-1}(x)}{dx} \right| \tag{2}$$

where $\left| \frac{df^{-1}(x)}{dx} \right|$ is the determinant of the Jacobian matrix for f^{-1}.

Details on the Jacobian Matrix

To understand how the Jacobian matrix is calculated, consider $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and $z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$. For these 2 dimensional distributions, we assume that they are related by f and f^{-1} such that

$$\begin{cases} x_1 = f_1(z), \\ x_2 = f_2(z). \end{cases} \quad \text{and} \quad \begin{cases} z_1 = f_1^{-1}(x), \\ z_2 = f_2^{-1}(x). \end{cases}$$

The Jacobian determinant for this example is:

$$\left| \frac{df^{-1}(x)}{dx} \right| = \det \begin{vmatrix} \frac{\partial f_1^{-1}}{\partial x_1}(x) & \frac{\partial f_1^{-1}}{\partial x_2}(x) \\ \frac{\partial f_2^{-1}}{\partial x_1}(x) & \frac{\partial f_2^{-1}}{\partial x_2}(x) \end{vmatrix}.$$

Remember that the determinant of a 2x2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is $ad - bc$. Therefore, the determinant for this example would be

$$\left| \frac{df^{-1}(x)}{dx} \right| = \frac{\partial f_1^{-1}}{\partial x_1}(x) \frac{\partial f_2^{-1}}{\partial x_2}(x) - \frac{\partial f_1^{-1}}{\partial x_2}(x) \frac{\partial f_2^{-1}}{\partial x_1}(x).$$

For normalizing flow model, we will be working with matrices much larger than 2x2. Therefore, under normal circumstances, the determinant would be rather difficult to calculate. To simplify the determinant calculation, flow models designs a special f^{-1} function such that the matrix is triangular.

Triangular matrices are matrices that have all 0 elements above or below the diagonal element. For example, they look like

$$U = \begin{matrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{matrix} \quad \text{or} \quad L = \begin{matrix} 7 & 0 & 0 \\ 8 & 9 & 0 \\ 10 & 11 & 12 \end{matrix}.$$

Triangular matrices are special because the determinants of these matrices are exceptionally easy to calculate. It is simply the product of the diagonal elements.

$$\det(U) = 1 \cdot 4 \cdot 6 = 24 \quad \text{and} \quad \det(L) = 7 \cdot 9 \cdot 12 = 756.$$

We will end up using this property later.

How does the Flow Model work?

Remember that $p(z)$ is assumed to be a **konwn** Isotropic Gaussian distribution. Given μ and σ^2, an isotropic Gaussian distribution has the density function:

$$p(z|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{1}{2\sigma^2}\|z - \mu\|^2\right). \tag{3}$$

Since Isotropic Gaussian is a well established distribution, it is very simple to sample from it. This implies that if we end up learning the function $x = f(z)$, we can generated a random sample from $z \sim p(z)$ and pass sample z into $f(z)$ to also generate a sample from $p(x)$.

Moreover, if we know f^{-1}, we also obtain $p(x)$ due to the prior relationship that was mentioned:

$$p(x) = p\left(z = f^{-1}(x)\right) \left|\frac{df^{-1}(x)}{dx}\right|. \tag{4}$$

Notice that if we have a sample $\hat{x}$ and we need to find $p(\hat{x})$, we can simply plug $\hat{x}$ into $\hat{z} = f^{-1}(\hat{x})$. Plugging $\hat{z}$ into $p(z)$, it enables us to indirectly calculate $p(\hat{x})$.

Understanding how we set up f^{-1}

In order for the model to work, everything has to be carefully designed. We start by understanding how f^{-1} is designed using an example. For this exmample, assume x and z both have n dimensions, e.g., for $n = 4$:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix}.$$

The flow model purposely designed a special inverse function f^{-1} to achieve $z = f^{-1}(x)$. What make f^{-1} special is that it guarantees the existence of f; having both f and f^{-1} is central for normalizing flow models to work.

More specifically, given x of 4 dimensions is transformed into z using the following structure diagram.

$$x = \begin{bmatrix} x_a \\ x_b \end{bmatrix} = \begin{matrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} & & \longrightarrow & & \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \\ \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} & \rightarrow & \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} e^{s(x_a)} + t(x_a) & \rightarrow & \begin{bmatrix} z_3 \\ z_4 \end{bmatrix} \end{matrix} = z.$$

Note that

- the vector x is divided into half. The top being x_a and the bottom as x_b.
- both $s(x_a)$ and $t(x_a)$ are just functions that takes the vector x_a and output a single number.
- f^{-1} can be more compactly written was

$$f^{-1}(x) = \begin{bmatrix} x_a \\ x_b \cdot e^{s(x_a)} + t(x_a) \end{bmatrix} = \begin{bmatrix} z_a \\ z_b \end{bmatrix} = z.$$

As mentioned previously, this function f^{-1} was purposely designed this way to guarantee the existence of f. Indeed, notice how we can find f easily with the following formula.

$$f(z) = \begin{bmatrix} z_a \\ \frac{z_b - t(z_a)}{e^{s(z_a)}} \end{bmatrix} = \begin{bmatrix} x_a \\ x_b \end{bmatrix} = x.$$

Here, $s(x_a)$ and $t(x_a)$ (or equivalently $s(z_a)$ and $t(z_a)$ in the inverse) are treated as constants, simplifying the inversion.

While $s(x_a)$ and $t(x_a)$ are constants in the context of the linear transformation of x_b, they themselves can be very complicated functions. Indeed, in practice, these functions are often modeled using neural networks to capture the complexity of the data distribution, while ensuring the overall transformation remains invertible.

Initially, we do not know $s(x)$ and $t(x)$. Similar to linear regression, they are parameterized by weights and our goal is to identify the best weights w that are most suitable for this situation. For this reason, we will now rewrite the equation with a new notation $s_w(x)$ and $t_w(x)$.

The Objective function for Normalizing Flows

Remember that Normalizing flow assumes that

1. $p(z)$ is an Isotropic Multivariate Gaussian distribution.
2. and that $p(z)$ is mathematically related to $p(x)$.

The objective of the flow model is the maximum likelihood objective where we want

$$\max_{f, f^{-1}} \prod_i^n p(x_i).$$

Due to the relationship between $p(z)$ and $p(x)$

$$p(x) = p(z) \left| \det \left(\frac{\partial f}{\partial z} \right)^{-1} \right|.$$

We can plug $p(x)$ into the maximum likelihood objective

$$\max_{f, f^{-1}} \prod_i^n p(z_i) \left| \det \left(\frac{\partial f}{\partial z} \right)^{-1} \right|.$$

Since the data we have are x_i and not z_i, we need to make a quick conversion with $z_i = f^{-1}(x_i)$, implying that our objective looks like

$$\max_w \prod_i^n p(f^{-1}(x)) \left| \det \left(\frac{\partial f^{-1}}{\partial x} \right) \right| \quad \text{where} \quad f^{-1}(x) = \begin{bmatrix} x_a \\ x_b \cdot e^{s_w(x_a)} + t_w(x_a) \end{bmatrix} = \begin{bmatrix} z_a \\ z_b \end{bmatrix} = z.$$

Notice that in the final formulation, we changed f, f^{-1} into w. This is the similar process with linear regression. Instead of looking for the function, we assume that the function is parameterized by a bunch of weights, and our goal is to use gradient descent to find the best weights that satisfies the objective.