Normalizing flow

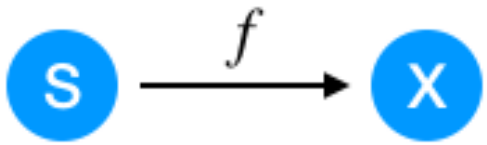
Presenter: Zhe Wang

https://qdata.github.io/deep2Read

Zhe Wang

201909

Data generating process:



Where $s_i, s_j, \forall i, j$ are independent.

- linear case
  $X = BS$, where B is a mixing matrix.
- nonlinear case
  $X = f(S)$, where f is an unknown but invertible function.

Given observations X, the theory of identifiability tells us when it is possible and to what extent we can recover S.

- Linear case:
  The theory of identifiability in linear ICA is well-developed.

- Nonlinear Case:
  The first breakthrought by Khemakhem et al. in 2019.
  Under relative mild assumption, we can recover the latent space distribution, up to a simple transformation in the latent space.
  Given X, we can recover $\hat{S} = AS + c$

The key requirement is that the generating process is conditioned on a variable which is observed along with the data.
Choices of U: class label, time index of a time series, or any other piece of information additional to the data.

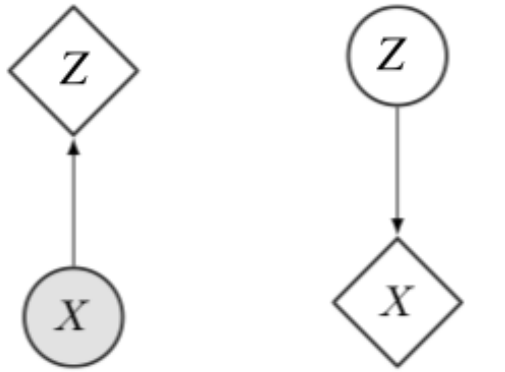
The task of nonlinear ICA is to:

- disentangle the data to recover the generating latent variables S
- recover the function f and its inverse

Suppose $S \in \mathbb{R}^D$ is a random variable, and $P_S : R^D \to R$ is a probability density function. There is a invertible transformation $f : R^D \to R^D$ which maps S to X and g is the inverse of f. The change of variable rule gives:

$$P_X(x) = P_S(g(x))|det\frac{\partial g(x)}{\partial x}|,$$

where $\frac{\partial g(x)}{\partial x}$ is the Jacobian matrix of function g at x.
Thus, we need to design f, g such that 1). the determinant of the Jacobian is trivially obtained. 2). the inverse $g = f^{-1}$ is also trivially obtained



(a) Inference (b) Sampling

Taking the log likelihood, we have:

$$\log P_X(x) = \log P_S(g(x)) + \log |det \frac{\partial g(x)}{\partial x}|,$$

Since we S_i are all mutually independent:

$$\log P_X(x) = \log \sum_{i=1}^{D} P_{s_i}(f(x)_i) + \log |det \frac{\partial f(x)}{\partial x}|.$$

NICE: Non-linear independent components estimation

Laurent Dinh, David Krueger, Yoshua Bengio

The architecture of the neural network is crucial to satisfy the requirement of f:

- obtain a family of bijections whose Jacobian determinant is tractable
- both f and g are easily computed.

The determinant of a composite function f is the product of its layers' Jacobian determinants. Thus, we define the operation for single layer.

$$det \frac{\partial g_1 \circ g_2}{\partial x} = det \frac{\partial g_1(g_2(x))}{\partial g_2(x)} det \frac{\partial g_2(x)}{\partial x}$$

General Coupling layer

let $X \in \mathbb{R}^D$, and $[I_1, I_2]$ is a partition of $[1, D]$, and $d = |I_1|$. Suppose m is any function defined on $\mathbb{R}^d$.

The general coupling layer is defined as:

$$\begin{aligned} S_{I_1} &= X_{I_1}, \\ S_{I_2} &= g(X_{I_2}; m(X_{I_1})), \end{aligned}$$

where $g : \mathbb{R}^{D-d} \times m(\mathbb{R}^d) \to \mathbb{R}^{D-d}$ is the coupling law.
The inverse of the function is:

$$\begin{aligned} X_{I_1} &= S_{I_1}, \\ X_{I_2} &= g^{-1}(S_{I_2}; m(S_{I_1})) \end{aligned}$$

We call such a transformation a coupling layer with coupling function m.

The Jacobian of the function is:

$$\frac{\partial s}{\partial x} = \begin{bmatrix} I_d, & 0 \\ \frac{\partial s_{I_2}}{\partial x_1}, & \frac{\partial s_{I_2}}{\partial x_2} \end{bmatrix},$$

That means $det\frac{\partial s}{\partial x} = det\frac{\partial s_{I_2}}{\partial x_2}$.
For simplicity, a good choice can be additive coupling law $g(a, b) = a + b$, so that:

$$\begin{aligned} S_{I_1} &= X_{I_1}, \\ S_{I_2} &= X_{I_2} + m(X_{I_1}), \end{aligned}$$

The inverse:

$$\begin{aligned} X_{I_1} &= S_{I_1}, \\ X_{I_2} &= S_{I_2} - m(S_{I_1})) \end{aligned}$$

The determinant is 1.

To improve the expressive power, we can compose several coupling layers. Since a coupling layer leaves part of its input unchanged, we need to exchange the role of the two subsets in the partition in alternating layers, so that the composition of two coupling layers modifies every dimension.

Disentanglement by nonlinear ICA with genearal incompressible-flow networks (GIN)

Peter Sorrenson, Carsten Rother, Ullrich Kothe, ICLR2020

Variable changing rule:

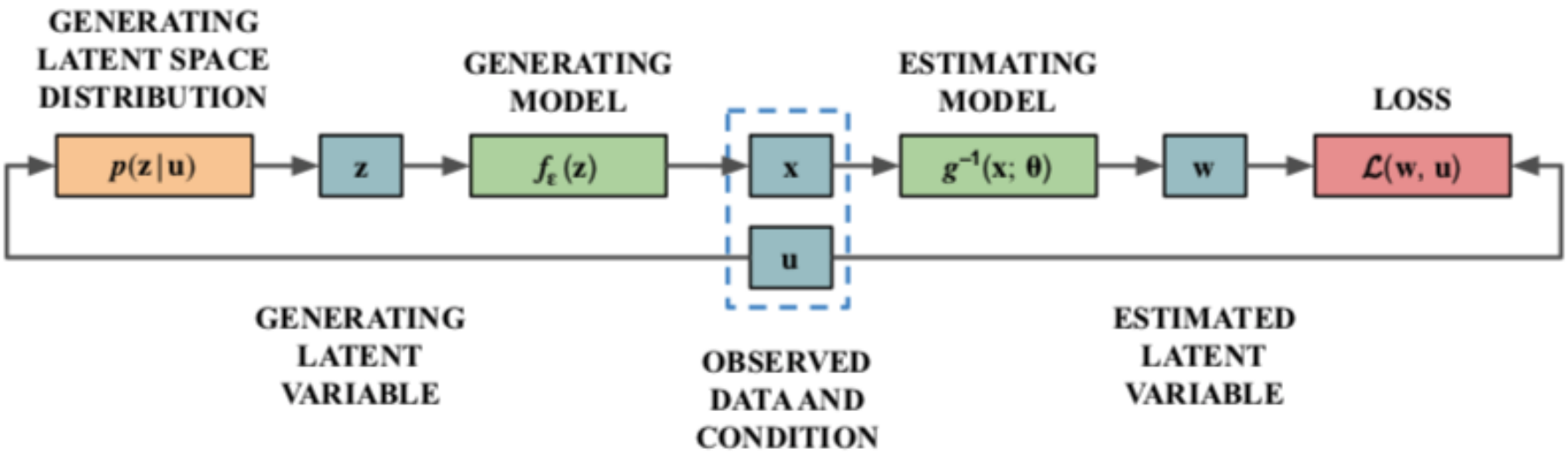
$$P_X(x) = P_S(g(x))|det\frac{\partial g(x)}{\partial x}|.$$

The density of a sample $x \in X$ can be computed from two components:

- the density of the inverse-transformed sample under this distribution
- the associated change in volume induced by the sequence of inverse transformations.

If the determinant of Jacobian can be calculated easily, and the pdf of s is tractable, the pdf of variable X can be derived explicitly.

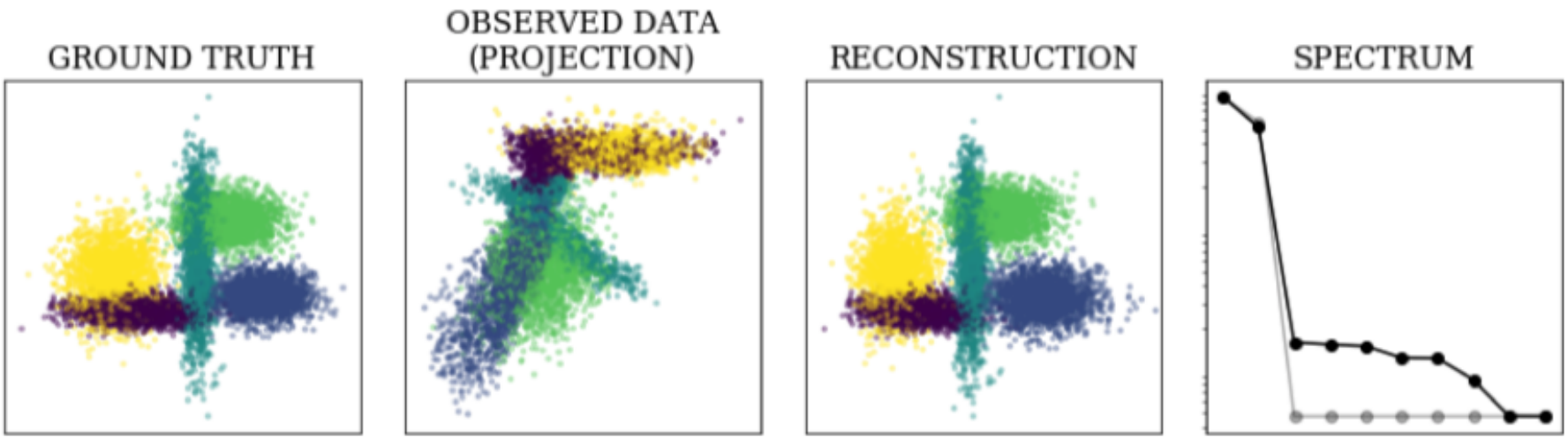
In order to truly recover f and g, from the requirement of identifiability of nonlinear ICA, there must be an additional observable variable U.

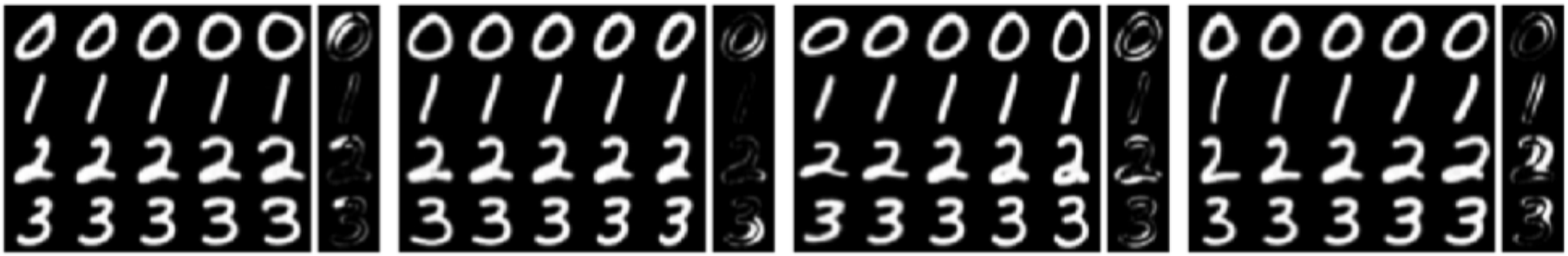


Model Architecture:

$$S_{I_1} = X_{I_1}, \tag{1}$$
$$S_{I_2} = X_{I_2} \cdot exp(C(X_{I_1})) + m(X_{I_1}) \tag{2}$$

where C is a scaling function. The log of determinant of the Jacobian will be the sum of $C(X_{I_1})$.





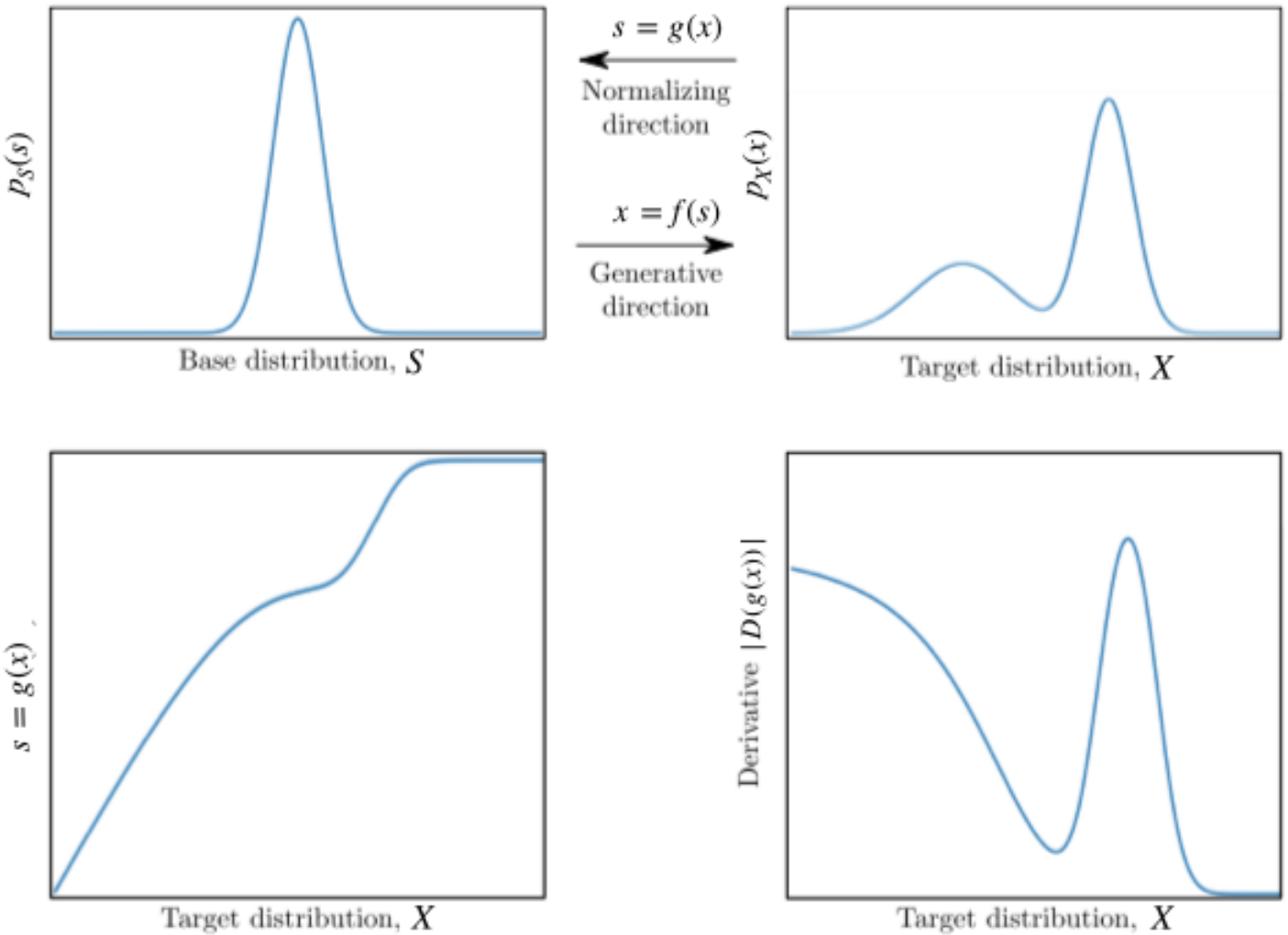
(a) Variable 1: upper width (b) Variable 8: lower width (c) Variable 3: height (d) Variable 4: bend

Normalizing flows: An introduction and review of current methods

Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker

A normalizing flow is a transformation of a simple probability distribution (e.g., a standard normal) into a more complex distribution by a sequence of invertible and differentiable mappings.

Target:



Normalizing flow should satisfy:

- be invertible. For sampling we need f while for computing likelihood we need g,
- be sufficiently expressive to model the distribution of interest,
- be computationally efficient in terms of f, g and determinant of Jacobian.

Types of flow:

- Elementwise Flows
- Linear Flows
- Coupling and Autoregressive Flows
- Residual Flows

Elementwise Flows:

A basic form of bijective non-linearity can be constructed given any bijective scalar function. Let $h : \mathbb{R} \to \mathbb{R}$ be a scalar valued bijection. Given $s = (s_1, s_2, \cdots, s_D)^T$

$f(s) = (h(s_1), h(s_2), \cdots, h(s_D))$ is also a bijection.

$f^{-1}(x) = (h^{-1}(x_1), h^{-1}(x_2), \cdots, h^{-1}(x_D))$ is the inverse of f.

Jacobian is the product of the absolute values of the derivatives of h.

In deep learning terminology, h, could be viewed as an "activation function", but the ReLU is not a proper choice.

Linear Flows:

Elementwise operations alone are insufficient as they cannot express any form of correlation between dimensions.

Linear mapping:
$f(s) = As + b$, where $A \in \mathbb{R}^{D \times D}, c \in \mathbb{R}^D$

The determinant of the Jacobian is $det(A)$, which can be computed in $O(D^3)$, also the inverse. Solution: Restricting the form of A.

- A is diagonal with nonzero diagonal entries. Inverse and determinant can be calculated in linear time. Limitation: cannot express correlation between dimensions.

- A is a triangular matrix. Determinant in $O(D)$, inversion in $O(D^2)$. Sensitive to the ordering of dimensions.
- Adding a permutation matrix whose determinant is 1. Can't be optimized and has to be fixed once initialized.
- A more general alternative is the use of orthogonal transformations.
- Instead of limiting the form of A, one can alternatively use LU decomposition:

$$f(s) = PLUs + b,$$

where L is lower triangular with ones on the diagonal, U is upper triangular with non-zero diagonal entries, and P is a permutation matrix. Determinant in $O(D)$ and inverse in $O(D^2)$

Coupling Flows:

let $S \in \mathbb{R}^D$, and $[I_1, I_2]$ is a partition of $[1, D]$, and $d = |I_1|$. Suppose m is an arbitrary function defined on $\mathbb{R}^d$. f is a bijective function.

$$
\begin{aligned}
X_{I_1} &= S_{I_1}, \\
X_{I_2} &= f(S_{I_2}; m(S_{I_1})),
\end{aligned}
$$

Since m can be arbitrarily complex, it is usually modelled as a neural network

- Affine coupling: $f(x_1; m(x_2)) = x_1 + m(x_2)$
- Nonlinear squared flow: $f(x_1; m(x_2)) = ax_1 + b + \dfrac{c}{1 + (dx + h)^2}$, where $m(x_2) = (a, b, c, d, h)$

Autoregressive flow:

Let $h(\cdot;\theta) : R \to R$ be a bijection parameterized by θ. Then an autoregressive model is a function $f : \mathbb{R}^D \to \mathbb{R}^D$, which outputs each entry of $x = f(s)$ conditioned on the previous entries of the input:

$$x_t = h(s_t; \theta_t(s_{1:t-1})),$$

where $s_{1:t} = (s_1, ..., s_t)$.

The Jacobian matrix of the autoregressive transformation f is triangular.

$$det(Df) = \prod_{t=1}^{D} \frac{\partial x_t}{\partial s_t}$$

However, the computation of the inverse is more challenging.
$s_1 = h^{-1}(x_1; \theta_1)$ and for any $t = 2, \cdots D, s_t = h^{-1}(x_t; \theta_t(s_{1:t1}))$.

Alternatively, one can use inverse autoregressive flow (IAF):

$$x_t = h(s_t; \theta_t(x_{1:t-1})),$$

Computation of the IAF is sequential and expensive, but the inverse is relatively efficient.