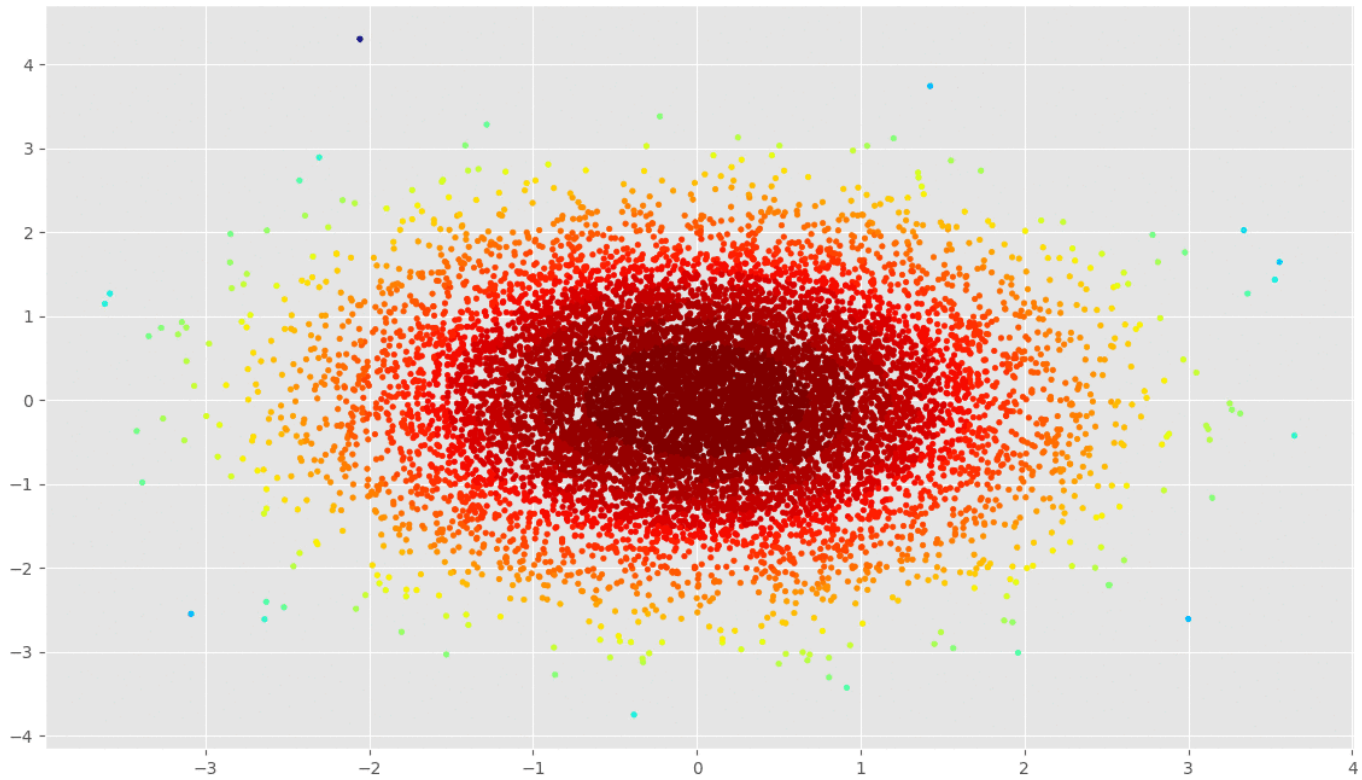
Normalizing Flows

November 5, 2020

Table of contents

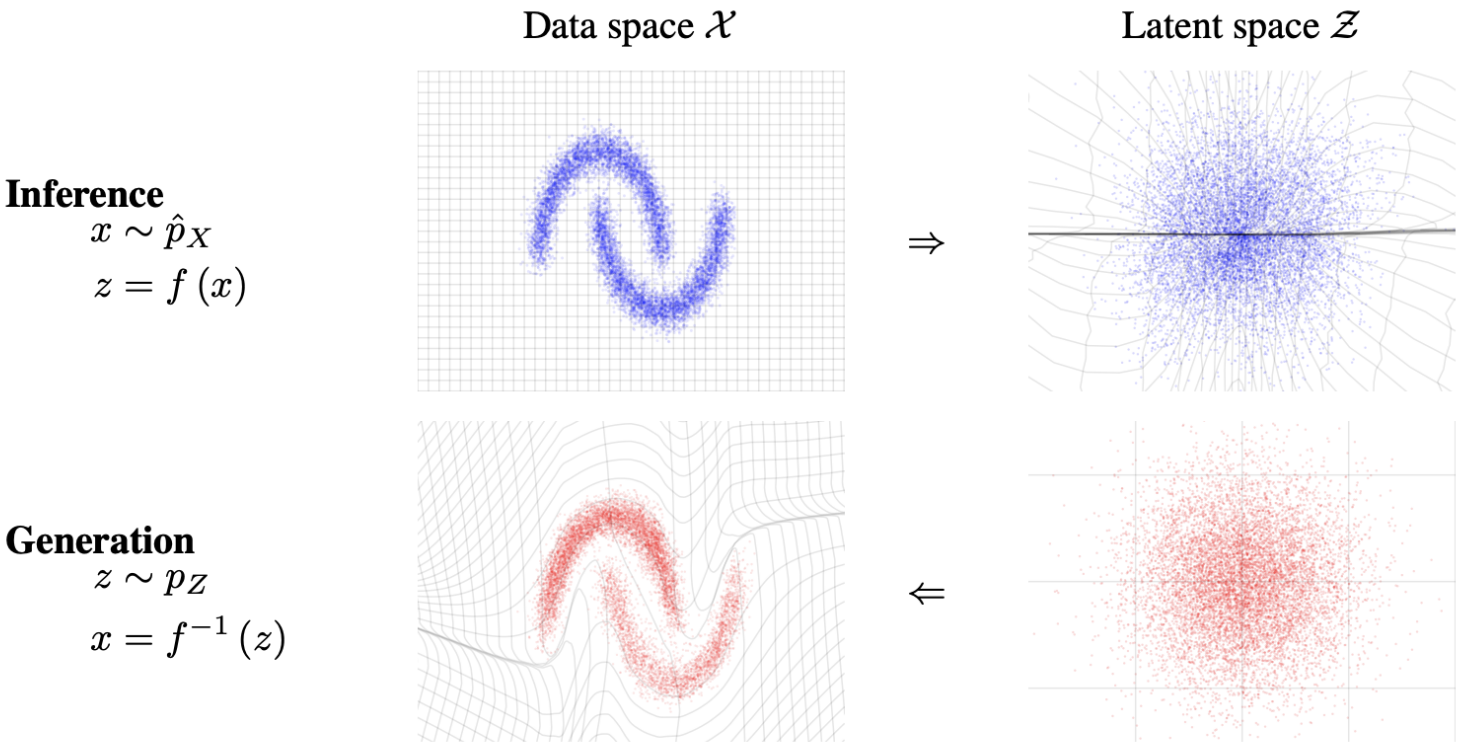
Overview

Main Idea



Normalizing flows provide a general mechanism for defining expressive probability distributions, only requiring the specification of a (usually simple) base distribution and a series of bijective transformations.

Animation Credit: Mike Slawinski, Ph.D., Cylance, Inc.



We learn an invertible mapping f between a data distribution $\widehat{p}_X$ and a latent distribution p_Z (typically a Gaussian). The technique allows one to perform anomaly detection and sample generation even for complex and high-dimensional distributions.

Image Credit: Dinh et al. (2017), ICLR.

Application: Sample Generation

On the left column, examples from a dataset. On the right column, samples from a normalizing flow model trained on the dataset. The datasets shown in this figure are in order: CIFAR-10, Imagenet (32×32), Imagenet (64×64), CelebA, LSUN (bedroom).

Application: Anomaly Detection in Cybersecurity

Application: Anomaly Detection in Cybersecurity

String stuffing attacks can fool ML-based Antivirus into thinking malware is good.

Malware	SHA256	Score Before	Score After
CoinMiner	1915126c27ba8566c624491bd2613215021cc2b28e5e6f3af69e9e994327f3ac	-826	884
Dridex	c94fe7b646b681ac85756b4ce7f85f4745a7b505f1a2215ba8b58375238bad10	-999	996
Emotet	b3be486490acd78ed37b0823d7b9b6361d76f64d26a089ed8fbd42d838f87440	-923	625
Gh0stRAT	eebff21def49af4e85c26523af2ad659125a07a09db50ac06bd3746483c89f9d	-975	998
Kovter	40050153dceec2c8fbb1912f8eeabe449d1e265f0c8198008be8b34e5403e731	-999	856
Nanobot	267912da0d6a7ad9c04c892020f1e5757edf9c4762d3de22866eb8a550bff81a	971	999
Pushdo	14c358cc64a929a1761e7ffeb76795e43ff5c8f6b9e21057bb98958b7fa11280	-999	999
Qakbot	869985182924ca7548289156cb500612a9f171c7e098b04550dbf62ab8f4ebd9	-998	991
Trickbot	954961fd69cbb2bb73157e0a4e5729d8fe967fdf18e4b691e1f76aeadbc40553	-973	774
Zeus	74031ad4c9b8a8757a712e14d120f710281027620f024a564cbea43ecc095696	-997	997

However, these Frankensteinian concoctions look anomalous in feature space.

Reference: `https://skylightcyber.com/2019/07/18/cylance-i-kill-you/`

Density estimation and change of variables

Consider a parametric function mapping continuous random variable X to continuous random variable Z

$$f_\theta : X \to Z$$
$$x \mapsto z$$

where x is an observed sample and z is a latent variable. Suppose p_Z is given.

Under some regularity conditions[1], the change of variables theorem gives us

$$\underbrace{p_X(x)}_{\textit{data space}} = \underbrace{p_Z(f_\theta(x)) \left| \det \frac{\partial f_\theta(x)}{\partial x} \right|}_{\textit{transformed latent space}}$$

[1] Importantly, f_θ must be a *diffeomorphism* – i.e. it must be invertible, and both it and its inverse must be differentiable

The **goal** of density estimation can be posed as follows: learn θ to model unknown data density p_X in terms of assumed latent variable density p_Z.

[2] E.g., straightforward computation of the determinant via LU decomposition is $\mathcal{O}(d^3)$, where d is the dimensionality od the data.

The **goal** of density estimation can be posed as follows: learn θ to model unknown data density p_X in terms of assumed latent variable density p_Z.

The **problem** is that for high-dimensional data, computing the Jacobian and determinant is in general computationally very expensive.[2]

[2] E.g., straightforward computation of the determinant via LU decomposition is $\mathcal{O}(d^3)$, where d is the dimensionality od the data.

The **goal** of density estimation can be posed as follows: learn θ to model unknown data density p_X in terms of assumed latent variable density p_Z.

The **problem** is that for high-dimensional data, computing the Jacobian and determinant is in general computationally very expensive.[2]

There appears to be a **conflict** between expressivity and efficiency.

[2] E.g., straightforward computation of the determinant via LU decomposition is $\mathcal{O}(d^3)$, where d is the dimensionality od the data.

Normalizing Flow

Definition

(Normalizing Flow) A *(normalizing) flow*, $f = h_\theta^1 \circ ... \circ h_\theta^K$, is a sequence of diffeomorphisms which maps an observed data point, x, to a latent state representation, z.

If we allow ourselves this abuse of notation[3]

$$h_\theta^0 := x$$
$$h_\theta^K := z$$

Then, since $\det \prod_i A_i = \prod_i \det A_i$, the likelihood becomes

$$p_X(x) = p_Z(f_\theta(x)) \prod_{k=1}^{K} \left| \det \frac{\partial h_\theta^k}{\partial h_\theta^{k-1}} \right| \tag{3.1}$$

[3] More generally, we will use the same notation to refer to the function itself as well as its evaluation at a point.

Real NVP

Idea

Real non-volume preserving flows (RNVP; Dinh et al., 2016) is a normalizing flow where each bijection in the sequence of bijections works in the following way.

- The d features of a random variable are partitioned into two groups.
- The first group is transformed by the identity transformation.
- The second group is transformed by a more complex operation whereby its values are scaled and translated (hence, an *affine transformation*) by factors created by applying neural networks to the features in the first group (hence, a *coupling transformation*).

Each such bijection is called an "affine coupling transformation" or "affine coupling layer" (ACL).

Definition

(Real NVP) A *real NVP* is a normalizing flow (Def. 1) where $f = h_\theta^1 \circ ... \circ h_\theta^K$ is structured such that:

$$h^{k+1} = b^k \odot h^k + (1 - b^k) \odot \left(h^k \odot \exp\left(s_\theta^k(b^k \odot h^k)\right) + t_\theta^k(b^k \odot h^k) \right)$$

where $b^1, ..., b^K$ is a sequence of binary masks, $\odot$ is the Hadamard product or element-wise product, and s and t stand for scale and translation.

Definition

(Affine Coupling Layer) An affine coupling layer is one element of the sequence of invertible transformations in a real NVP; i.e. it is h_k for some $k \in \{1, ..., K\}$ in Def. 2.

Example

If random variables are D dimensional, and $b^k := [1, \ldots 1, 0, \ldots 0]$, where the 0 entries begin at the $(d_k + 1)$st element, then the affine coupling layer is given by

$$h^{k+1}_{1:d_k} = h^k_{1:d_k}$$
$$h^{k+1}_{d_k+1:D} = h^k_{d_k+1:D} \odot \exp\left(s^k_\theta(h^k_{1:d_k})\right) + t^k_\theta(h^k_{1:d_k})$$

Note that the real NVP allows for efficient computation of the determinant of the Jacobians, since

$$\frac{\partial h_\theta^{k+1}}{\partial h_\theta^k} = \begin{pmatrix} \mathbb{I}_{d_k} & 0 \\ \frac{\partial h_{d_k+1:D}^{k+1}}{\partial h_{1:d_k}^k} & \operatorname{diag}\left(\exp\left(s_\theta(h_{1:d_k}^k) \right) \right) \end{pmatrix}$$

The bottom left term can be arbitrarily complex; we don't have to compute it, since the determinant of a triangular matrix is the product of the diagonals:

$$\det \frac{\partial h_\theta^{k+1}}{\partial h_\theta^k} = \exp\left(\sum_j s_\theta^k \left(h_{1:d_k}^k \right)_j \right)$$

So, by Equation 3.1, the log likelihood with real NVP normalizing flow applied to a single data sample, x, is

$$\log p_X(x) = \log p_Z(f_\theta(x)) + \sum_k \sum_j s_\theta^k \left(h_{1:d_k}^k\right)_j$$

So, by Equation 3.1, the log likelihood with real NVP normalizing flow applied to a single data sample, x, is

$$\log p_X(x) = \log p_Z(f_\theta(x)) + \sum_k \sum_j s_\theta^k \left(h_{1:d_k}^k\right)_j$$

And the log likelihood applied for a collection of samples, assumed *i.i.d.*, is the sum of individual log likelihoods.

So, by Equation 3.1, the log likelihood with real NVP normalizing flow applied to a single data sample, x, is

$$\log p_X(x) = \log p_Z(f_\theta(x)) + \sum_k \sum_j s_\theta^k \left(h_{1:d_k}^k\right)_j$$

And the log likelihood applied for a collection of samples, assumed *i.i.d.*, is the sum of individual log likelihoods.

This evaluation is linear in the number of samples, number of features, and number of layers.