

生成模型(三):Flow-based model



理想主义者

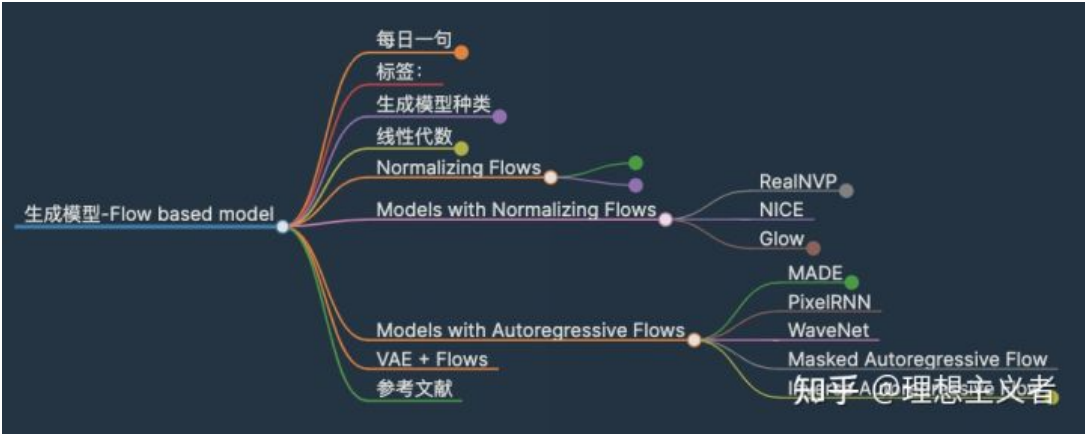
6 人赞同了该文章

本文译自：[Flow-based Deep Generative Models](#)

每日一句

Think in the morning. Act in the noon. Eat in the evening. Sleep in the night. — William Blake

本文大纲如下：



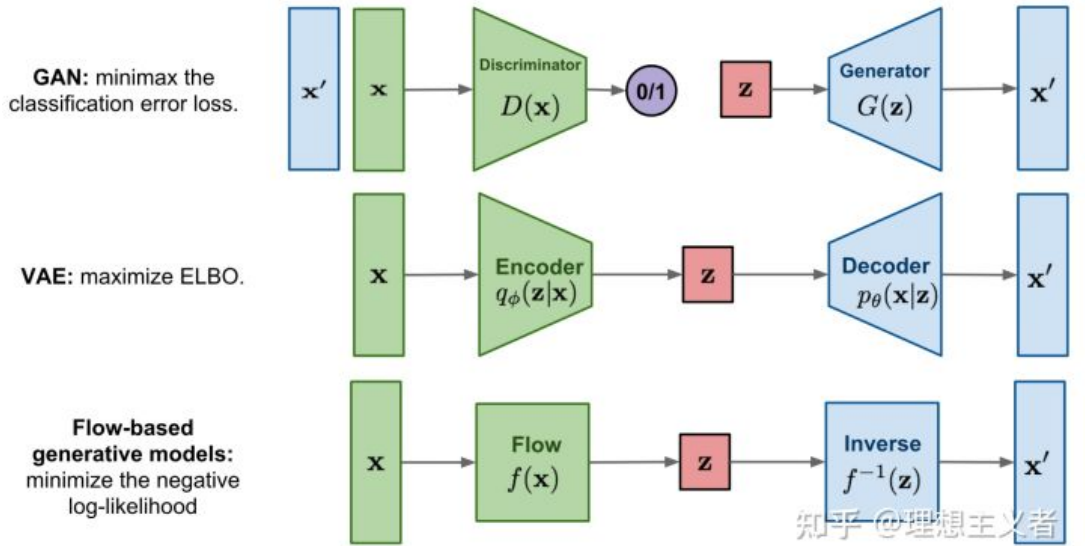
到目前为止，已经介绍了[[生成模型-GAN]]和[[生成模型-VAE]]。它们都没有明确地学习真实数据的概率密度函数 $p(\mathbf{x})$ （其中 $\mathbf{x} \in \mathcal{D}$ ），因为很难。以带有潜变量的生成模型为例， $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$ 几乎无法计算，因为要遍历隐变量 \mathbf{z} 的所有可能值， 这点是难以做到的。

基于流的深度生成模型`normalizing flows` 解决了这个难题，这是一个强大的密度估计的统计工具。对 $p(\mathbf{x})$ 的良好估能有效地完成许多下游任务：对未观察到的但现实的新数据点进行采样（数据生成），预测未来事件的罕见性（密度估计），推断潜在变量，填补不完整的数据样本，等等。

生成模型种类

下面是对GAN、VAE和基于流的生成模型之间区别的一个快速总结。

1. *生成对抗网络*。GAN提供了一个聪明的解决方案，将数据生成这一无监督的学习问题建模为有监督的问题。鉴别器模型学习从生成器模型产生的虚假样本中区分出真实数据。两个模型同时训练，进行最大最小博弈。
2. *变量自动编码器*。VAE通过最大化证据下限（ELBO），不明确地优化了数据的对数可能性。
3. *基于流的生成模型*。基于流的生成模型是由一连串的可逆变换构建的。与其他两种不同，该模型明确地学习了数据分布 $p(\mathbf{x})$ ，因此损失函数只是负对数似然。



线性代数



雅各布矩阵和行列式

给定一个将 n 维输入向量 \mathbf{x} 映射到 m 维输出向量的函数， $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ ，这个函数的所有一阶偏导的矩阵被称为 Jacobian 矩阵， \mathbf{J} ，其中第 i 行和第 j 列是 $\mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j}$ 。

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

行列式是一个实数，作为一个平方矩阵中所有元素的函数来计算。请注意，行列式只存在于方形矩阵中。行列式的绝对值可以被认为衡量矩阵的乘法对空间的扩张或收缩程度的标准。

一个 $n \times n$ 矩阵 M 的行列式是:

$$\det M = \det \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \sum_{j_1 j_2 \dots j_n} (-1)^{\tau(j_1 j_2 \dots j_n)} a_{1j_1} a_{2j_2} \dots a_{nj_n}$$

其中 $j_1 j_2 \dots j_n$ 的下标是集合 $1, 2, \dots, n$ 的所有置换，所以总共有 $n!$ 项； $\tau(.)$ 表示置换的特征。

一个正方形矩阵 M 的行列式可以检测它是否可逆。如果 $\det(M) = 0$ ，那么 M 是不可逆的（一个具有线性依赖行或列的奇异矩阵；或者任何行或列都是 0）；否则，如果 $\det(M) \neq 0$ ， M 是可逆的。

乘积的行列式等同于行列式的乘积： $\det(AB) = \det(A) \det(B)$ 。

Change of Variable Theorem

让我们回顾一下变量变化定理，从一个单变量的情况开始。给定一个随机变量 z 及其已知的概率密度函数 $z \sim \pi(z)$ ，我们想用单射函数 $x = f(z)$ 构造一个新的随机变量。该函数 f 是可逆的，所以 $z = f^{-1}(x)$ 。现在的问题是如何推断新变量的未知概率密度函数 $p(x)$ ？

$$\int p(x) dx = \int \pi(z) dz = 1; \text{ Definition of probability distribution.}$$
$$p(x) = \pi(z) \left| \frac{dz}{dx} \right| = \pi(f^{-1}(x)) \left| \frac{df^{-1}}{dx} \right| = \pi(f^{-1}(x)) |(f^{-1})'(x)|$$

根据定义，积分 $\int \pi(z) dz$ 是无限个宽度为 Δz 的矩形之和。这样一个矩形在位置 z 的高度就是密度函数 $\pi(z)$ 的值。当我们替代变量时， $z = f^{-1}(x)$ 得到 $\frac{\Delta z}{\Delta x} = (f^{-1}(x))'$ 和 $\Delta z = (f^{-1}(x))' \Delta x$ 。这里 $|(f^{-1}(x))'|$ 表示分别在两个不同的变量坐标 z 和 x 中定义的矩形面积的比率。

多变量版本也有类似的格式:

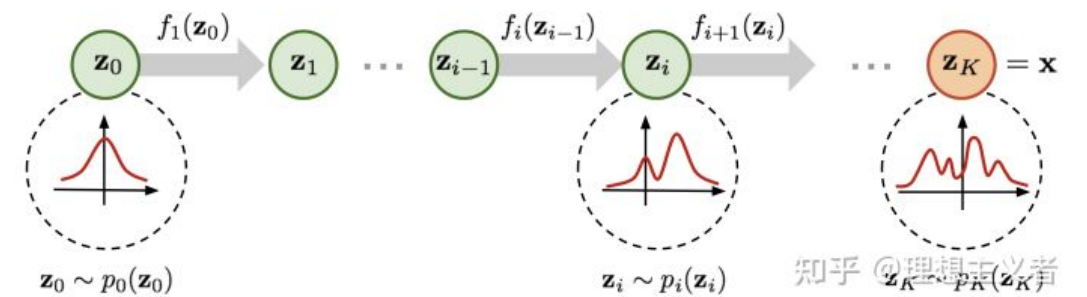
$$\mathbf{z} \sim \pi(\mathbf{z}), \mathbf{x} = f(\mathbf{z}), \mathbf{z} = f^{-1}(\mathbf{x})$$
$$p(\mathbf{x}) = \pi(\mathbf{z}) \left| \det \frac{d\mathbf{z}}{d\mathbf{x}} \right| = \pi(f^{-1}(\mathbf{x})) \left| \det \frac{df^{-1}}{d\mathbf{x}} \right|$$

其中 $\det \frac{\partial f}{\partial \mathbf{z}}$ 是函数 f 的雅各布行列式。

Normalizing Flows

由于我们需要在深度学习模型中运行反向传播，希望后验概率 $p(\mathbf{z}|\mathbf{x})$ 可以简单有效地计算出导数。这也是为什么高斯分布经常被用于隐变量生成模型，尽管现实世界中的大多数分布比高斯分布要复杂得多。

Normalizing Flows (NF) 模型，用于更好和更强大的分布近似。NF 通过应用一连串的可逆变换函数将一个简单的分布转变成一个复杂的分布。通过一连串的转变，我们根据变量变化定理反复用新的变量来替代，最终得到最终目标变量的概率分布。



如图所示：



$$p_i(\mathbf{z}_i) = p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{df_i^{-1}}{d\mathbf{z}_i} \right|$$

然后把方程转换成 \mathbf{z}_i 的函数，这样就可以做推理。

$$\begin{aligned} p_i(\mathbf{z}_i) &= p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{df_i^{-1}}{d\mathbf{z}_i} \right| \\ &= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \left(\frac{df_i}{d\mathbf{z}_{i-1}} \right)^{-1} \right| && \text{; According to the inverse func theorem.} \\ &= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|^{-1} && \text{; According to a property of Jacobians of invertible func.} \\ \log p_i(\mathbf{z}_i) &= \log p_{i-1}(\mathbf{z}_{i-1}) - \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right| \end{aligned}$$

关于反函数定理的说明。如果 $y = f(x)$ 和 $x = f^{-1}(y)$ ，有：

$$\frac{df^{-1}(y)}{dy} = \frac{dx}{dy} = \left(\frac{dy}{dx}\right)^{-1} = \left(\frac{df(x)}{dx}\right)^{-1}$$

关于逆函数的雅各布式的说明。可逆矩阵的行列式是行列式的倒数： $\det(M^{-1}) = (\det(M))^{-1}$,因为 $\det(M) \det(M^{-1}) = \det(M \cdot M^{-1}) = \det(I) = 1$ 。

鉴于概率密度函数链，我们知道每一对连续变量之间的关系。我们可以一步步展开输出 \mathbf{x} 的方程，直到追溯到初始分布 \mathbf{z}_0 。

$$\begin{aligned} \mathbf{x} &= \mathbf{z}_K = f_K \circ f_{K-1} \circ \dots \circ f_1(\mathbf{z}_0) \\ \log p(\mathbf{x}) &= \log \pi_K(\mathbf{z}_K) = \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\ &= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log \left| \det \frac{df_{K-1}}{d\mathbf{z}_{K-2}} \right| - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\ &= \dots \\ &= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^K \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right| \end{aligned}$$

随机变量 $\mathbf{z}_i = f_i(\mathbf{z}_{i-1})$ 所经过的路径是流，由连续分布 π_i 形成的全链被称为归一化流。根据方程中的计算要求，转换函数 f_i 应满足两个特性。

- 可逆
- 雅各比行列式容易计算

Models with Normalizing Flows

有了归一化流，输入数据的精确对数似然 $\log p(\mathbf{x})$ 就变得可操作了。因此，基于流的生成模型的训练标准就是训练数据集 \mathcal{D} 上的负对数似然（NLL）。

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x})$$

RealNVP

RealNVP（Real-valued Non-Volume Preserving Dinh等人，2017）模型通过堆叠一连串的可逆双射变换函数来实现归一化流。在每个双射函数 $f: \mathbf{x} \mapsto \mathbf{y}$ ，被称为仿射耦合层(affine coupling layer)，输入维度被分成两部分。

- 前 d 维度保持不变。
- $d + 1$ 到 D 维度，经历一个仿射变换，系数和截距参数都是前 d 维度的函数。

$$\begin{aligned} \mathbf{y}_{1:d} &= \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} &= \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d}) \end{aligned}$$

其中 $s(\cdot)$ 和 $t(\cdot)$ 是比例函数和平移函数，都是进行 $\mathbb{R}^d \mapsto \mathbb{R}^{D-d}$ 映射。 \odot 操作是点乘。

现在让检查一下这个变换是否满足流变换的两个基本属性。

- [x] 条件1：可逆

$$\begin{cases} \mathbf{y}_{1:d} &= \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} &= \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d}) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} &= \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} &= (\mathbf{y}_{d+1:D} - t(\mathbf{y}_{1:d})) \odot \exp(-s(\mathbf{y}_{1:d})) \end{cases}$$

- [x] 条件2: 雅各比行列式容易计算

雅各布矩阵是一个下三角矩阵。

$$\mathbf{J} = \begin{bmatrix} \mathbb{I}_d & \mathbf{0}_{d \times (D-d)} \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{x}_{1:d}} & \text{diag}(\exp(s(\mathbf{x}_{1:d}))) \end{bmatrix}$$

因此，行列式只是对角线上的项的乘积。



由于a.计算 f^{-1} 不需要计算 s 或 t 的逆，2. 计算雅各布行列式不涉及计算 s 或 t ，这些函数可以是任意复杂的；即 s 和 t 都可以用深度神经网络来建模。

在仿射耦合层中，一些维度保持不变。为了确保所有的输入都有机会被改变，该模型在每一层中颠倒了顺序，使不同的成分保持不变。按照这样的交替模式，在一个转化层中保持相同的单元集在下一个转化层中总是被修改。BN被发现有助于训练具有非常深的耦合层堆栈的模型。

此外，RealNVP可以在多尺度架构下工作，为大的输入建立一个更有效的模型。多尺度架构将几种采样操作应用于normal affine layers，包括spatial checkerboard pattern masking, squeezing operation, 和 channel-wise masking。

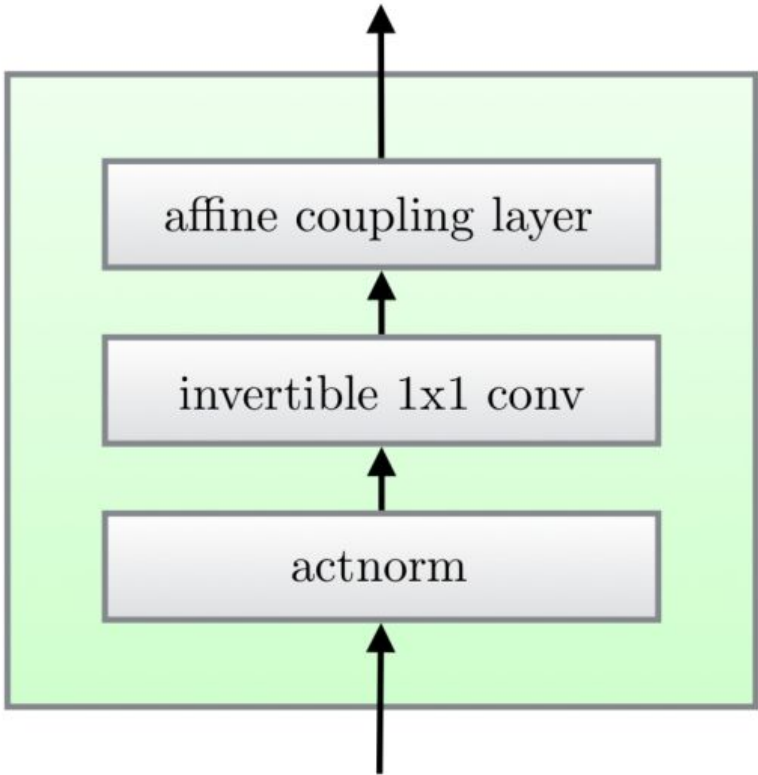
NICE

NICE（"Non-linear Independent Component EstimationDinh, et al. 2015）模型是realnvp的前身。NICE中的变换是不含系数项的仿射耦合层，被称为加性耦合层。

$$\begin{cases} \mathbf{y}_{1:d} &= \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} &= \mathbf{x}_{d+1:D} + m(\mathbf{x}_{1:d}) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} &= \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} &= \mathbf{y}_{d+1:D} - m(\mathbf{y}_{1:d}) \end{cases}$$

Glow

Glow（GlowKingma and Dhariwal, 2018）模型扩展了之前的模型NICE和RealNVP，并通过用可逆的1x1卷积取代通道排序上的反向互换操作来简化架构。



知乎 @理想主义者

Glow中，有三个子步骤。

1. 激活归一化（简称actnorm） 它对每个通道用比例和偏置参数进行仿射转换，类似于批处理归一化，但是对mini批处理大小为1进行处理。参数是可训练的，但初始化后，第一批的数据在actnorm后的平均值为0，标准差为1。
2. 可逆的1x1 conv 在RealNVP流程的各层之间，通道的顺序是相反的，这样所有的数据维度都有机会被改变。输入和输出通道数量相等的1×1卷积可以对通道进行任何变换。

假设有一个输入 $h \times w \times c$ 张量的可逆1x1卷积 \mathbf{h} 与一个大小为 $c \times c$ 的权重矩阵 \mathbf{W} 。输出是一个 $h \times w \times c$ 张量，标记为 $\mathbf{f} = \text{conv2d}(\mathbf{h}; \mathbf{W})$ 。为了应用变量变化规则，我们需要计算雅各布行列式 $|\det \partial \mathbf{f} / \partial \mathbf{h}|$ 。

1x1卷积的输入和输出都可以看作是一个大小为 $h \times w$ 的矩阵。 \mathbf{x}_{ij} ($i = 1, \dots, h, j = 1, \dots, w$) 在 \mathbf{h} 中是一个 c 通道的向量，每个item分别乘以权重矩阵 \mathbf{W} ，得到输出矩阵中的相应条目 \mathbf{y}_{ij} 。每个条目的导数为 $\partial \mathbf{x}_{ij} \mathbf{W} / \partial \mathbf{x}_{ij} = \mathbf{W}$ ，总共有 $h \times w$ 这样的item。

$$\log \left| \det \frac{\partial \text{conv2d}(\mathbf{h}; \mathbf{W})}{\partial \mathbf{h}} \right| = \log(|\det \mathbf{W}|^{h \cdot w}) = h \cdot w \cdot \log |\det \mathbf{W}|$$

1x1卷积的逆取决于逆矩阵 \mathbf{W}^{-1} 。由于权重矩阵相对较小，矩阵行列式和逆的计算量仍在控制之中。

1. 仿生耦合层 该设计与RealNVP相同。

See Section 3.1.			
Invertible 1×1 convolution. $\mathbf{W} : [c \times c]$. See Section 3.2.	$\forall i, j : \mathbf{y}_{i,j} = \mathbf{W} \mathbf{x}_{i,j}$	$\forall i, j : \mathbf{x}_{i,j} = \mathbf{W}^{-1} \mathbf{y}_{i,j}$	$h \cdot w \cdot \log \det(\mathbf{W}) $ or $h \cdot w \cdot \text{sum}(\log \mathbf{s})$ (see eq. (10))
Affine coupling layer. See Section 3.3 and (Dinh et al., 2014)	$\mathbf{x}_a, \mathbf{x}_b = \text{split}(\mathbf{x})$ $(\log \mathbf{s}, \mathbf{t}) = \text{NN}(\mathbf{x}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{y}_a = \mathbf{s} \odot \mathbf{x}_a + \mathbf{t}$ $\mathbf{y}_b = \mathbf{x}_b$ $\mathbf{y} = \text{concat}(\mathbf{y}_a, \mathbf{y}_b)$	$\mathbf{y}_a, \mathbf{y}_b = \text{split}(\mathbf{y})$ $(\log \mathbf{s}, \mathbf{t}) = \text{NN}(\mathbf{y}_b)$ $\mathbf{s} = \exp(\log \mathbf{s})$ $\mathbf{x}_a = (\mathbf{y}_a - \mathbf{t})/\mathbf{s}$ $\mathbf{x}_b = \mathbf{y}_b$ $\mathbf{x} = \text{concat}(\mathbf{x}_a, \mathbf{x}_b)$	$\text{sum}(\log(\mathbf{s}))$

Models with Autoregressive Flows

自回归约束是建立顺序数据模型的一种方式， $\mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_D]$ ：每个输出只取决于过去观察到的数据，但不取决于未来的数据。换句话说，观察到 \mathbf{x}_i 的概率是以 $\mathbf{x}_1, \dots, \mathbf{x}_{i-1}$ 为条件的，这些条件概率的乘积给我们观察到完整序列的概率。

$$p(\mathbf{x}) = \prod_{i=1}^D p(\mathbf{x}_i | \mathbf{x}_1, \dots, \mathbf{x}_{i-1}) = \prod_{i=1}^D p(\mathbf{x}_i | \mathbf{x}_{1:i-1})$$

如何对条件密度进行建模? 可以是单变量高斯作为 $\mathbf{x}_{1:i-1}$ 的函数计算，或者是以 $\mathbf{x}_{1:i-1}$ 为输入的多层神经网络。

如果将归一化流中的流变换定为自回归模型：向量变量中的每个维度都以之前的维度为条件，这就是自回归流。

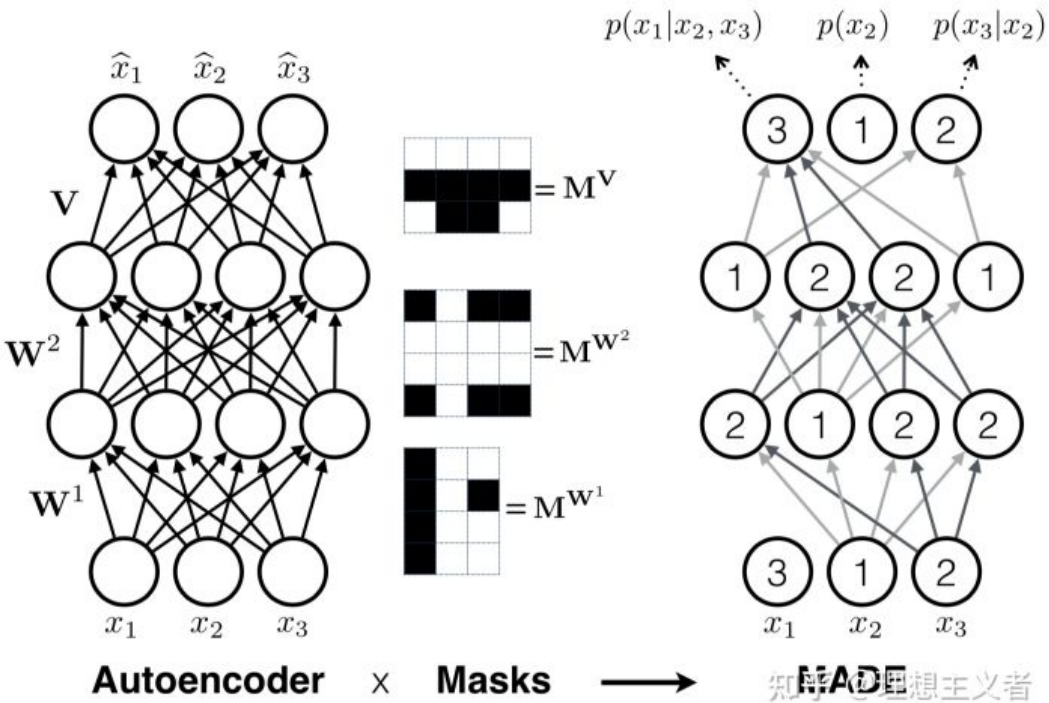
本节从几个经典的自回归模型（MADE，PixelRNN，WaveNet）开始，然后我们深入研究自回归流模型（MAF和IAF）。

MADE

MADE（Masked Autoencoder for Distribution EstimationGermain等人，2015）可以在自动编码器中有效地利用自回归特性。当使用自动编码器预测条件概率时，MADE不是将不同观察窗口 D 次的输入输入自动编码器，而是通过乘以二进制掩码矩阵来去除某些隐藏单元的贡献，这样，每个输入维度在单次中只从以前的维度中按给定的顺序进行重建。

在一个多层全连接神经网络中，例如，有 L 隐层，其权重矩阵为 $\mathbf{W}^1, \dots, \mathbf{W}^L$ ，输出层的权重矩阵为 \mathbf{V} 。输出 $\hat{\mathbf{x}}$ 有 $\hat{\mathbf{x}}_i = p(\mathbf{x}_i | \mathbf{x}_{1:i-1})$ 。在没有任何掩码的情况下，通过层的计算:

$$\begin{aligned} \mathbf{h}^0 &= \mathbf{x} \\ \mathbf{h}^l &= \text{activation}^l(\mathbf{W}^l \mathbf{h}^{l-1} + \mathbf{b}^l) \\ \hat{\mathbf{x}} &= \sigma(\mathbf{V} \mathbf{h}^L + \mathbf{c}) \end{aligned}$$



为了消除层间的一些连接，可以简单地将每个权重矩阵逐元乘以一个二进制掩码矩阵。每个隐藏节点被分配一个随机的连接整数; 介于 1 和 $D - 1$ 之间; l 层中第 k 单元的分配值用 m_k^l 表示。二进制掩码矩阵是通过对两层中的两个节点的数值进行元素比较而确定的。

$$\begin{aligned} \mathbf{h}^l &= \text{activation}^l((\mathbf{W}^l \odot \mathbf{M}^{\mathbf{W}^l}) \mathbf{h}^{l-1} + \mathbf{b}^l) \\ \hat{\mathbf{x}} &= \sigma((\mathbf{V} \odot \mathbf{M}^{\mathbf{V}}) \mathbf{h}^L + \mathbf{c}) \\ M_{k',k}^{\mathbf{W}^l} &= \mathbf{1}_{m_{k'}^l \geq m_k^{l-1}} = \begin{cases} 1, & \text{if } m_{k'}^l \geq m_k^{l-1} \\ 0, & \text{otherwise} \end{cases} \\ M_{d,k}^{\mathbf{V}} &= \mathbf{1}_{d \geq m_k^L} = \begin{cases} 1, & \text{if } d \geq m_k^L \\ 0, & \text{otherwise} \end{cases} \end{aligned}$$

当前层中的单元只能与前一层中数字相同或更小的其他单元相连，这种类型的依赖性很容易通过网络传播到输出层。一旦数字被分配给所有的神经元和层，输入维度的排序就被固定了。条件概率也



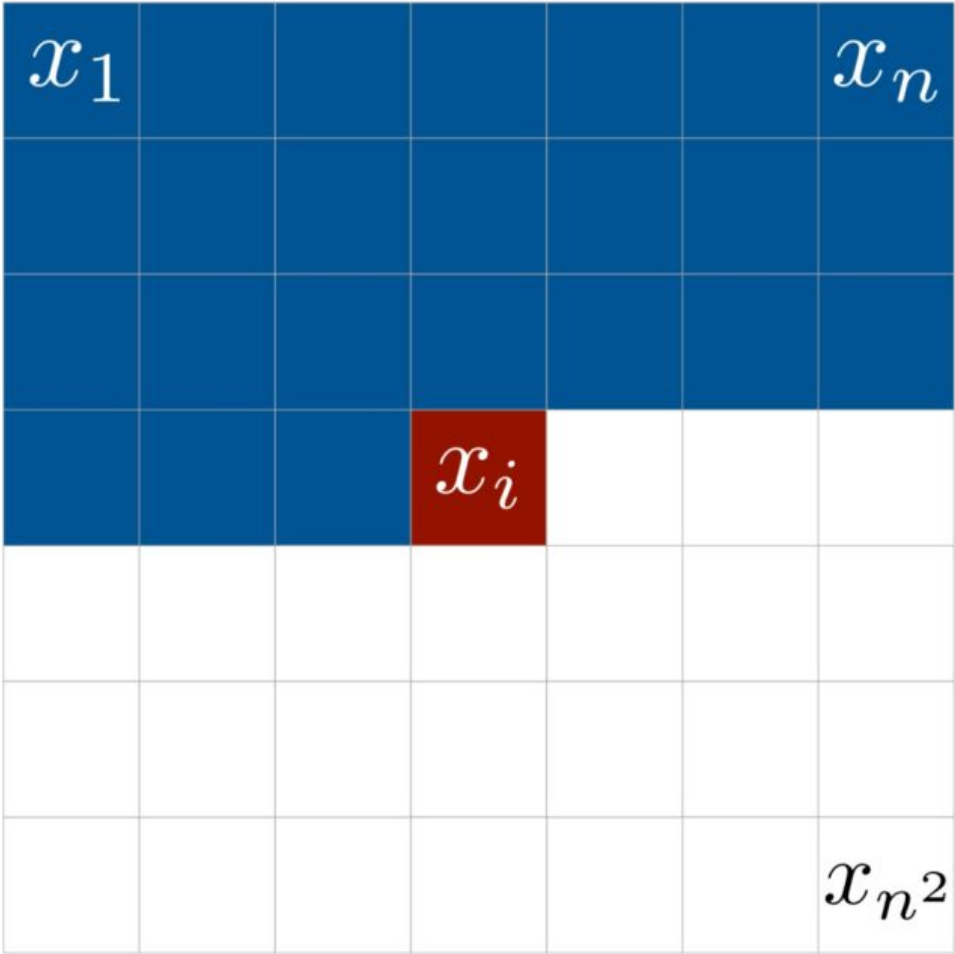
MADE训练可以通过以下方式进一步促进:

- 顺序无关的训练：对输入维度进行shuffle，使MADE能够对任何任意的顺序进行建模；可以在运行时创建自回归模型的集合。
- 连通性无关的训练：为了避免模型被特定的连通性模式约束所束缚，对每个训练的minibatch重新取样 m_k^l 。

PixelRNN

PixelRNN（PixelRNNOord等人，2016）是一个用于图像的深度生成模型。图像是一次生成一个像素，每个新的像素都是以之前像素为条件进行采样。

考虑一个大小为 $n \times n$ 的图像， $\mathbf{x} = \{x_1, \dots, x_{n^2}\}$ ，该模型从左上角开始生成像素，从左到右，从上到下。

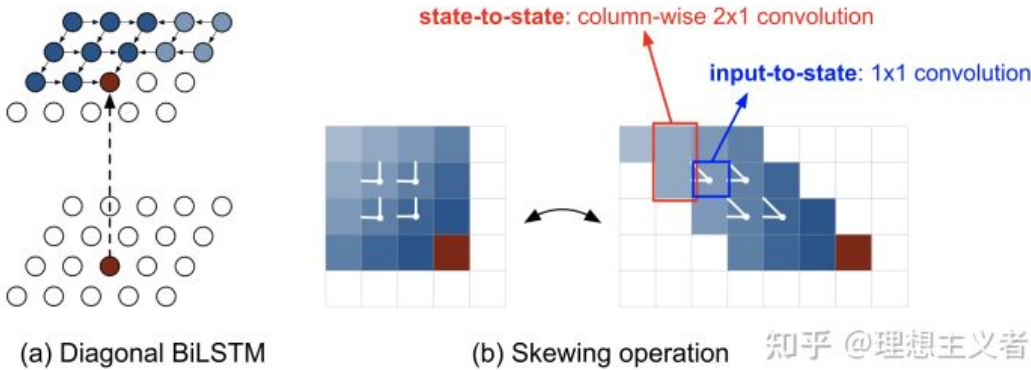


Context

知乎 @理想主义者

每个像素 x_i 都是通过以过去条件（在同一行中，它上面的像素或它左边的像素）下的概率分布中取样的。这种背景的定义看起来很随意，因为视觉注意力如何被关注到图像上是比较灵活的。但具有如此强烈假设的生成模型有不可思议的记过。

一个可以捕获整个上下文的实现是 Diagonal BiLSTM。首先，应用skewing操作，将输入特征图的每一行相对于前一行偏移一个位置，这样每一行的计算就可以被并行化。然后，LSTM的状态是相对于当前像素和左边的像素计算的。



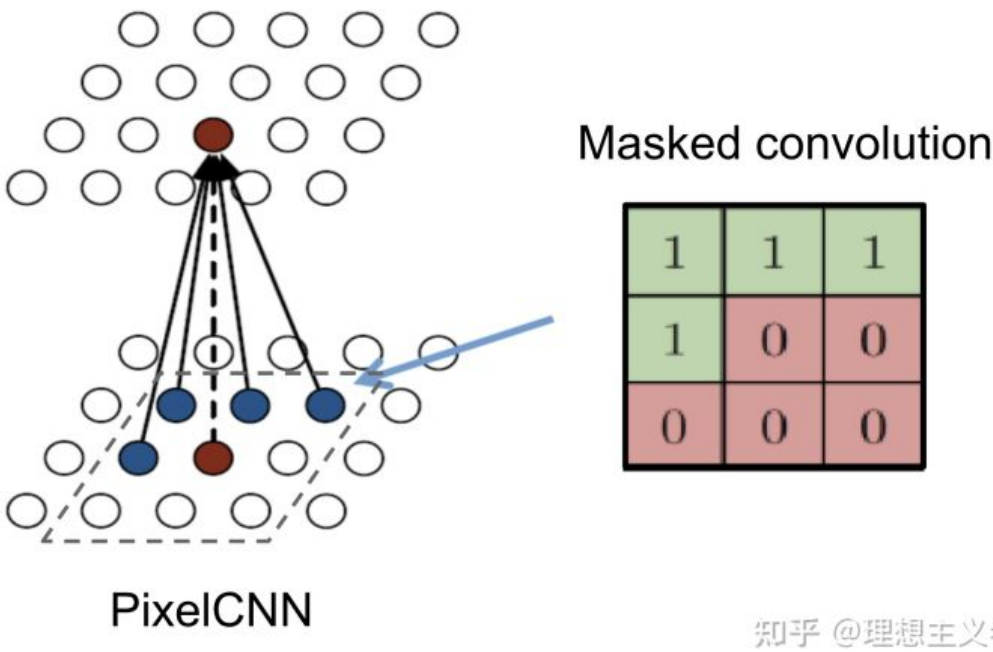
知乎 @理想主义者

$$\begin{aligned} [\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] &= \sigma(\mathbf{K}^{ss} \otimes \mathbf{h}_{i-1} + \mathbf{K}^{is} \otimes \mathbf{x}_i) && \sigma \text{ is tanh for g, but otherwise sigmoid; } \otimes \text{ is convolution operation.} \\ \mathbf{c}_i &= \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i && \odot \text{ is elementwise product.} \\ \mathbf{h}_i &= \mathbf{o}_i \odot \tanh(\mathbf{c}_i) \end{aligned}$$

其中 \otimes 表示卷积操作， \odot 是元素间的乘法。输入到状态的分量 \mathbf{K}^{is} 是一个1x1的卷积，而状态到状态的循环分量是用一个2x1的内核的列向卷积 \mathbf{K}^{ss} 计算的。



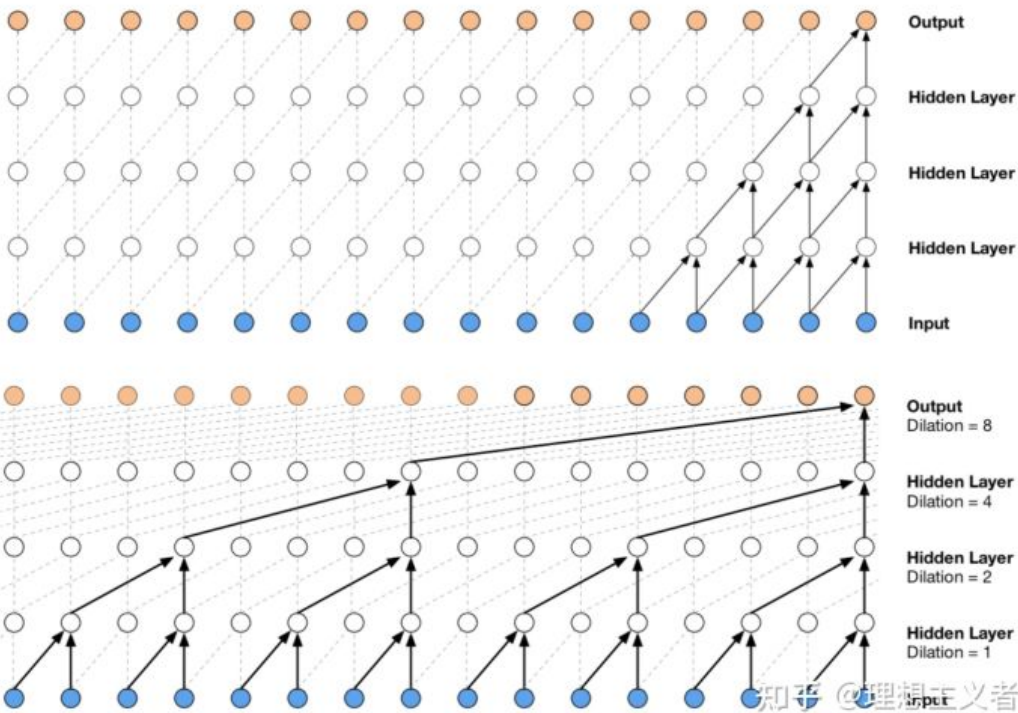
看不到未来的上下文，类似于MADE。这个卷积版本被称为PixelCNN。



WaveNet

WaveNet(Van Den Oord, et al. 2016)与PixelCNN非常相似，但应用于一维音频信号。WaveNet由一叠因果卷积(causal convolution)组成，这是一种注重排序的卷积操作：在某个时间戳的预测只能消耗过去观察到的数据，对未来没有依赖性。在PixelCNN中，因果卷积是通过掩码卷积核实现的。在WaveNet中，因果卷积只是将输出移到未来的若干个时间戳，使输出与最后的输入元素保持一致。

卷积层的一个很大的缺点是接受域的大小非常有限。输出很难依赖于几百或几千个时间段之前的输入，这是对长序列建模的关键要求。因此，WaveNet采用了扩张卷积（dilated convolution）。



WaveNet使用门控激活单元作为非线性层，因为人们发现它在一维音频数据建模时明显比ReLU效果好。残差连接应用在门控激活后。

$$\mathbf{z} = \tanh(\mathbf{W}_{f,k} \otimes \mathbf{x}) \odot \sigma(\mathbf{W}_{g,k} \otimes \mathbf{x})$$

其中 $\mathbf{W}_{f,k}$ 和 $\mathbf{W}_{g,k}$ 分别是第 k 层的卷积滤波器和矩阵。

Masked Autoregressive Flow

Masked Autoregressive Flow（MAF Papamakarios等人，2017）是一种归一化流，其中转换层被构建为自回归神经网络。MAF与后面介绍的逆自回归流（IAF）非常相似。

给定两个随机变量， $\mathbf{z} \sim \pi(\mathbf{z})$ 和 \mathbf{x} 的概率密度函数 $p(\mathbf{z})$ 已知，MAF 目标是学习 $p(\mathbf{x})$ 。MAF根据过去的维度 $\mathbf{x}_{1:i-1}$ 生成 \mathbf{x}_i 。条件概率是 \mathbf{z} 的仿射变换，其中系数和截距是 \mathbf{x} 的观察部分的函数。

数据生成，产生一个新的 \mathbf{x} ：

$$\mathbf{x}_i \sim p(\mathbf{x}_i | \mathbf{x}_{1:i-1}) = \mathbf{z}_i \odot \sigma_i(\mathbf{x}_{1:i-1}) + \mu_i(\mathbf{x}_{1:i-1}), \text{ 其中 } \mathbf{z} \sim \pi(\mathbf{z})$$

密度估计，给定一个已知的 \mathbf{x} 。

$$p(\mathbf{x}) = \prod_{i=1}^D p(\mathbf{x}_i | \mathbf{x}_{1:i-1})$$

与MAF类似，逆自回归流（**IAF**Kingma等人，2016）将目标变量的条件概率也建模为自回归模型，但流向相反，从而实现更高效的采样过程。

首先，让我们reverse MAF中的仿射变换。

$$z_i = \frac{x_i - \mu_i(\mathbf{x}_{1:i-1})}{\sigma_i(\mathbf{x}_{1:i-1})} = -\frac{\mu_i(\mathbf{x}_{1:i-1})}{\sigma_i(\mathbf{x}_{1:i-1})} + x_i \odot \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})}$$

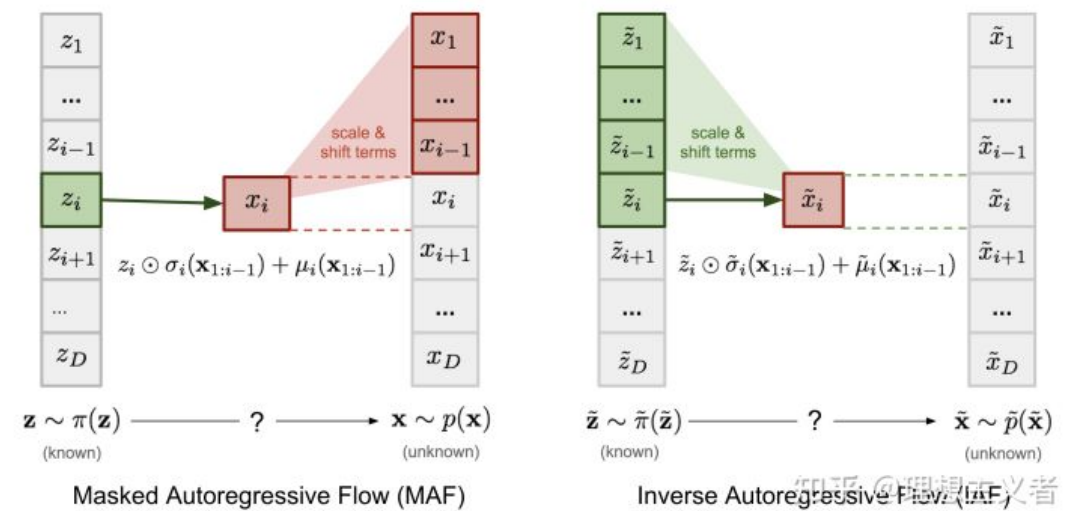
假设：

$$\begin{aligned} \tilde{\mathbf{x}} &= \mathbf{z}, \tilde{p}(\cdot) = \pi(\cdot), \tilde{\mathbf{x}} \sim \tilde{p}(\tilde{\mathbf{x}}) \\ \tilde{\mathbf{z}} &= \mathbf{x}, \tilde{\pi}(\cdot) = p(\cdot), \tilde{\mathbf{z}} \sim \tilde{\pi}(\tilde{\mathbf{z}}) \\ \tilde{\mu}_i(\tilde{\mathbf{z}}_{1:i-1}) &= \tilde{\mu}_i(\mathbf{x}_{1:i-1}) = -\frac{\mu_i(\mathbf{x}_{1:i-1})}{\sigma_i(\mathbf{x}_{1:i-1})} \\ \tilde{\sigma}(\tilde{\mathbf{z}}_{1:i-1}) &= \tilde{\sigma}(\mathbf{x}_{1:i-1}) = \frac{1}{\sigma_i(\mathbf{x}_{1:i-1})} \end{aligned}$$

可得：

$$\tilde{x}_i \sim p(\tilde{x}_i|\tilde{\mathbf{z}}_{1:i}) = \tilde{z}_i \odot \tilde{\sigma}_i(\tilde{\mathbf{z}}_{1:i-1}) + \tilde{\mu}_i(\tilde{\mathbf{z}}_{1:i-1}), \text{ 其中 } \tilde{\mathbf{z}} \sim \tilde{\pi}(\tilde{\mathbf{z}})$$

IAF打算在 $\tilde{\mathbf{x}}$ 已知的情况下估计 $\tilde{p}_i(\tilde{\mathbf{z}})$ 的概率密度函数。逆流也是一个自回归仿射变换，与MAF相同，但系数和截距是来自已知分布 $\tilde{p}_i(\tilde{\mathbf{z}})$ 的观察变量的自回归函数。



各个元素 \tilde{x}_i 的计算并不相互依赖，所以它们很容易并行化（使用MADE只需一次）。对于一个已知的 $\tilde{\mathbf{x}}$ 的密度估计并不高效，因为我们必须按顺序恢复 \tilde{z}_i 的值， $\tilde{z}_i = (\tilde{x}_i - \tilde{\mu}_i(\tilde{\mathbf{z}}_{1:i-1}))/\tilde{\sigma}_i(\tilde{\mathbf{z}}_{1:i-1})$ ，因此一共是D次。

	初始分布	目标分布	模型	生成	密度估计
MAF	$\mathbf{z} \sim \pi(\mathbf{z})$	$\mathbf{x} \sim p(\mathbf{x})$	$x_i = z_i \odot \sigma_i(\mathbf{x}_{1:i-1}) + \mu_i(\mathbf{x}_{1:i-1})$	序列，慢	点，快
IAF	$\tilde{\mathbf{z}} \sim \tilde{\pi}(\tilde{\mathbf{z}})$	$\tilde{\mathbf{x}} \sim \tilde{p}(\tilde{\mathbf{x}})$	$\tilde{x}_i = \tilde{z}_i \odot \tilde{\sigma}_i(\tilde{\mathbf{z}}_{1:i-1}) + \tilde{\mu}_i(\tilde{\mathbf{z}}_{1:i-1})$	序列，慢	点，快

VAE + Flows

在VAE中，如果我们想把后验 $p(\mathbf{z}|\mathbf{x})$ 建模为一个更复杂的分布，而不是简单的高斯分布。直观地说，我们可以使用归一化流来转换高斯，以获得更好的密度近似。然后，编码器将预测一组系数和截距 (μ_i, σ_i) ，它们都是输入 \mathbf{x} 的函数。具体详见 [f-VAEs: Improve VAEs with Conditional Flows](#)

参考文献

[1] Danilo Jimenez Rezende, and Shakir Mohamed. [“Variational inference with normalizing flows.” ICML 2015.](#)

[2] [Normalizing Flows Tutorial, Part 1: Distributions and Determinants by Eric Jang.](#)

[3] [Normalizing Flows Tutorial, Part 2: Modern Normalizing Flows by Eric Jang.](#)

[4] [Normalizing Flows by Adam Kosiorek.](#)

[5] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. [“Density estimation using Real NVP.” ICLR 2017.](#)

[6] Laurent Dinh, David Krueger, and Yoshua Bengio. [“NICE: Non-linear independent components estimation.” ICLR 2015 Workshop track.](#)

[7] Diederik P. Kingma, and Prafulla Dhariwal. [“Glow: Generative flow with invertible](#)



[9] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. “Pixel recurrent neural networks." ICML 2016.

[10] Diederik P. Kingma, et al. “Improved variational inference with inverse autoregressive flow." NIPS. 2016.

[11] George Papamakarios, Iain Murray, and Theo Pavlakou. “Masked autoregressive flow for density estimation." NIPS 2017.

[12] Jianlin Su, and Guang Wu. “f-VAEs: Improve VAEs with Conditional Flows." arXiv:1809.05861 (2018).

[13] Van Den Oord, Aaron, et al. “WaveNet: A generative model for raw audio." SSW. 2016.

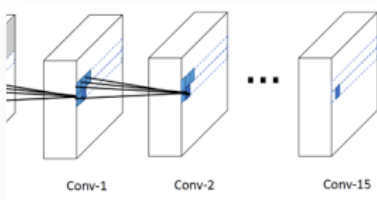
编辑于 2022-04-25 09:41

生成模型

文章被以下专栏收录

无监督学习

推荐阅读



自回归模型 - PixelCNN

deeph... 发表于deeph...

生成模型(四):扩散模型

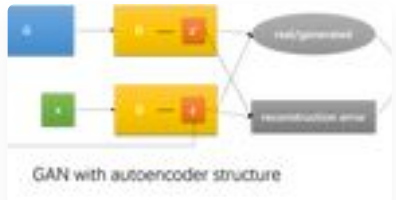
本文译自：What are Diffusion Models? & Generative Modeling by Estimating Gradients of the Data Distribution （Part）
每日一句 Think like a man of action; act like a man of ...

理想主义者 发表于无监督学习



flow-based生成模型

liured



关于生成模型的一些小思考

Gapen... 发表于学术兴趣小...

2 条评论

切换为时间排序

写下你的评论...

 Mat

基础概念补充非常便于理解了，写这个真的费心思了

 赞

 知乎用户

写的真好

 赞

