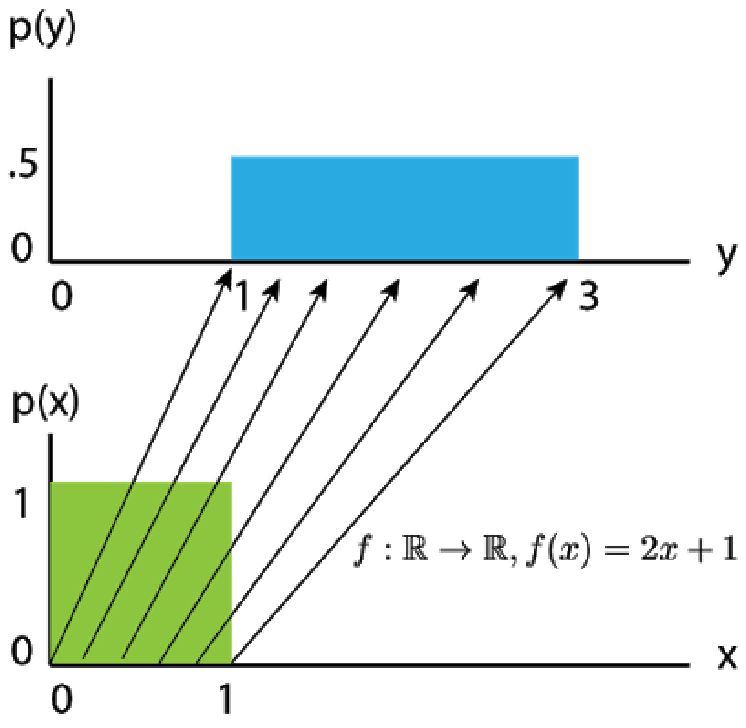
Normalizing Flow

Basics

> Change of variable rule : Change of variables, change of volume, In mathematics, a **change of variables** is a basic technique used to simplify problems in which the original variables are replaced with functions of other variables.

一维变量线性变换

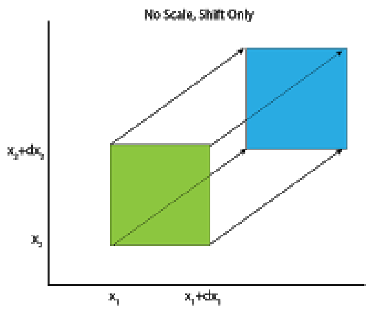
变量X服从$Uniform(0,1)$分布，变量$Y=f(X)=2X+1$，则X与Y的分布分别如下，即变量改变，对应概率分布改变。

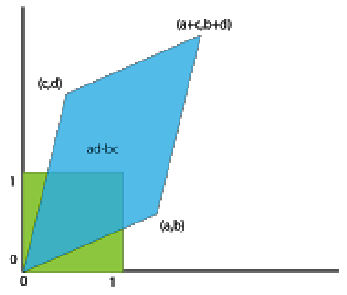


二维变量线性变换

考虑二维变量，例如以下变换矩阵，变量的分布范围以及概率分布发生改变

$$A=\begin{bmatrix}dx_1\\dx_2\end{bmatrix},Y=XB=\begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}a & b\\c & d\end{bmatrix}$$





变换后平行四边形面积等于$ad-bc$，正好等于变换矩阵B的行列式$detB=ad-bc$

1. 根据坐标计算，面积等于两个向量(a,b)、(c,d)内积等于$ad-bc$
2. $S=|a||b|sin(\theta)=|a||b|sin(\alpha-\beta)=|a||b|(sin\alpha cos\beta-sin\beta cos\alpha)=\sqrt{a^2+b^2}\sqrt{c^2+d^2}\frac{da}{\sqrt{a^2+b^2}}\frac{bc}{\sqrt{c^2+d^2}}=ad-bc$

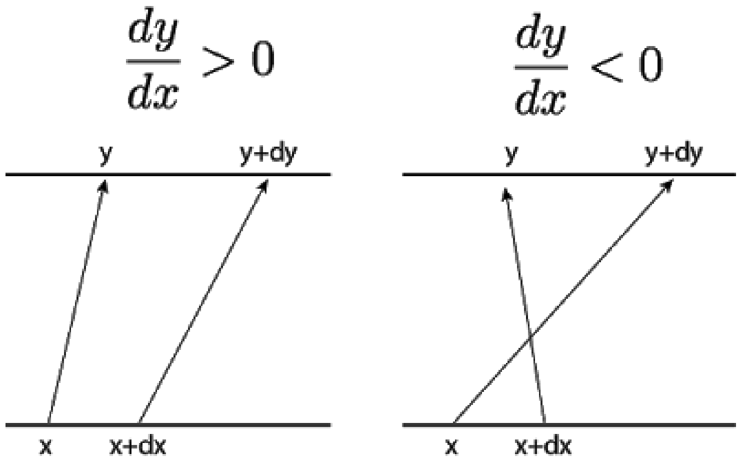
多维变量线性变换

三维的情况时，“转换为平行四边形”就对应为“转换为平行六面体”；

更高维n的情况，“转换为平行2n维体”。

行列式的道理也还是如此，**线性变换后的体积，正好对应于变换矩阵的行列式。**

小结



根据概率分布积分为1，则有

$$p(x)dx = p(y)dy$$
$$p(y) = p(x)\frac{dx}{dy}$$

左半边的图代表一个局部增函数，右边则是局部减函数。

前提：$p(x)$因为dx带来的局部变化一定要等于$p(y)$随着dy的局部变化，我们只关心变化的量，而不关心变化的方向，则有

$$p(y) = p(x)|\frac{dx}{dy}|$$

其中$y = f(x)$,x与y均表示n维变量,$f(.)$为可逆变换且逆函数为

$$g(.) = f^{-1}(.)$$
$$x = f^{-1}(y) = g(f(x))$$

根据上文推理，可知

$$\frac{dy}{dx} = detJ(B) = detJ(f(x)),\text{注意}\frac{dy}{dx}\text{为}n \times n\text{维矩阵}$$

由于$f(.)$为可逆变换，因此

$$\frac{dx}{dy} = detJ(f^{-1}(y))$$
$$|\frac{dx}{dy}| = |detJ(f^{-1}(y))|$$

则变量y的概率分布可表示为

$$
\begin{aligned}
p(y) = p(x) \cdot |detJ(f^{-1}(y))| \\
= p(f^{-1}(y)) \cdot |detJ(f^{-1}(y))|
\end{aligned}
$$

为了提高计算稳定性，应用时通常使用对数（log or ln）概率密度

$$log(p(y)) = log(p(f^{-1}(y)) + log(|detJ(f^{-1}(y))|)$$

非线性变换

根据小结所述，雅可比行列式即可覆盖所有线性与非线性变换的情况，即把线性变换当做一种特殊的映射。

上述二维变换的介绍只是便于理解，可以直接看小结推导，推理恒成立。即在假设条件下，以下等式恒成立。

$$
\begin{gathered}
p(y) = p(f^{-1}(y)) \cdot |detJ(f^{-1}(y))| \\
log(p(y)) = log(p(f^{-1}(y)) + log(|detJ(f^{-1}(y))|)
\end{gathered}
$$

引申

根据Jacobian Matrix和行列式性质，

$$
\begin{gathered}
\frac{dx}{dy} = J(f^{-1}(y)), \frac{dy}{dx} = J(f(x)), \frac{dy}{dx} \cdot \frac{dx}{dy} = I \\
\text{则} det\frac{dx}{dy} = \frac{1}{det\frac{dy}{dx}} = (det\frac{dy}{dx})^{-1}, \\
|detJ(f^{-1}(y))| = |detJ(f(x))|^{-1}
\end{gathered}
$$

因此，

$$
\begin{aligned}
p(y) &= p(f^{-1}(y)) \cdot |detJ(f^{-1}(y))| \\
&= p(x) \cdot |detJ(f(x))|^{-1} \\
lnp(y) &= lnp(x) - ln|detJ(f(x))|
\end{aligned}
$$

Normalizing Flows

使用n个可逆映射变换，施加到某个已知的概率密度分布，变换到一个新的（复杂的）概率分布，称为nflows

假设多个不同的可逆映射：

$$f_n : \mathbb{R}^d \to \mathbb{R}$$

对初始随机变量做多次可逆映射，映射嵌套

$$z_0 \sim q_0(z_0)$$
$$z_K = f_K \circ f_{K-1} \circ \ldots \circ f_1(z_0)$$

则多次变换后的变量服从以下分布

$$z_K \sim q_K(z_K) = q_0(z_0) \cdot \prod_{k=1}^{K} |det \frac{\partial f_k}{\partial z_{k-1}}|^{-1}$$
$$log(q_K(z_K)) = log(q_0(z_0)) - \sum_{k=1}^{K} log|det \frac{\partial f_k}{\partial z_{k-1}}|$$
$$ln(q_K(z_K)) = ln(q_0(z_0)) - \sum_{k=1}^{K} ln|det \frac{\partial f_k}{\partial z_{k-1}}|$$
$$ln(q_K(z_K)) = ln(q_0(z_0)) - \sum_{k=1}^{K} ln|det \frac{\partial f_k^{-1}}{\partial z_k}|^{-1} = ln(q_0(z_0)) + \sum_{k=1}^{K} ln|det \frac{\partial f_k^{-1}}{\partial z_k}|$$

Planar Flow

在平面空间对原分布进行变换，适用于低维情况，映射函数：

$$f(z) = z + \mu h(\omega^T \cdot z + b)$$

where μ, ω and b are trainable parameters, $h(.)$ is element-wise non-linear transform.

logdet-Jacobian与行列式（基于matrix determinant lemma）：

$$\psi(z) = h^{'}(\omega^T \cdot z + b) \cdot \omega$$
$$|det \frac{\partial f}{\partial z}| = |det(I + \mu \cdot \psi(z)^T)| = |1 + \mu^T \cdot \psi(z)|$$

Radial Flow

在球形空间对原分布进行变换，适用于低维情况，映射函数与行列式：

$$f(z) = z + \beta \cdot h(\alpha, r) \cdot (z - z_0)$$
$$where\ r = |z - z_0|, h(\alpha, r) = \frac{1}{\alpha + r}$$
$$|det \frac{\partial f}{\partial z}| = [1 + \beta \cdot h(\alpha, r)]^{d-1}[1 + \beta \cdot h(\alpha, r) + \beta \cdot h^{'}(\alpha, r) \cdot r]$$

> Variational Inference with Normalizing Flows

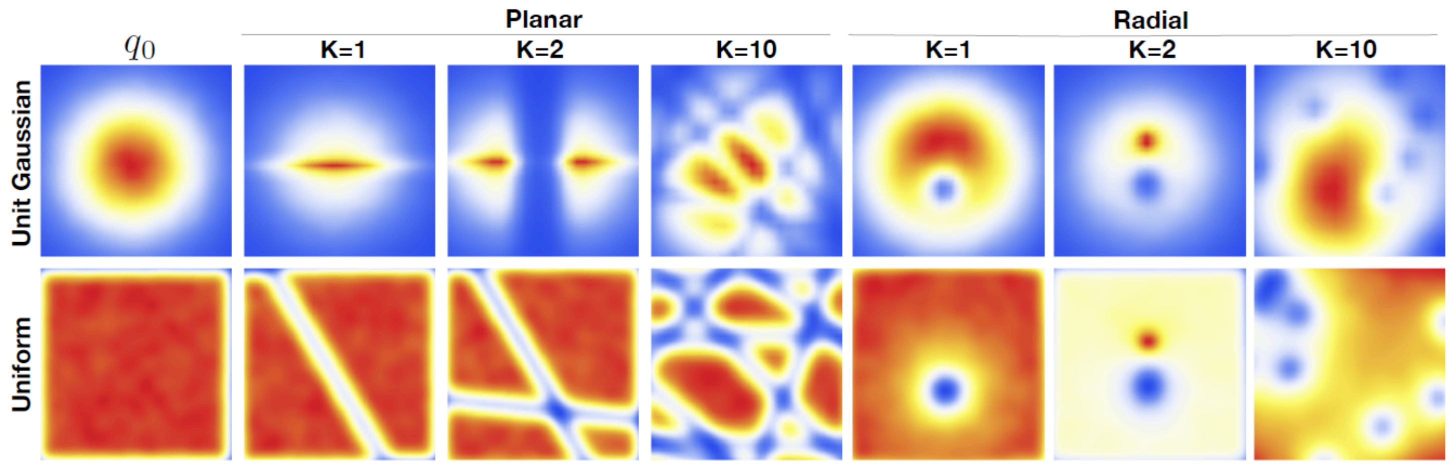


Figure 1. Effect of normalizing flow on two distributions.

Summary

- Planar and radial flows are suitable for low-dimension issues
- It is hard to compute Inverse matrix and Jacobian determinant.

Coupling Flow

假设$x \in \mathbb{R}^D$, I_1, I_2将$\mathbb{R}^D$分为两部分，其中$I_1 \in \mathbb{R}^{(1,d)}$, $I_2 \in \mathbb{R}^{(d+1,D)}$, m is a coupling function defined on $\mathbb{R}^{(1,d)} \rightarrow \mathbb{R}^{(d+1,D)}$

图中$x_{plain} \leftarrow x_{I_2}, x_{key} \leftarrow x_{I_1}$ and $y_{cipher} \leftarrow y_{I_2}, y_{key} \leftarrow y_{I_1}$

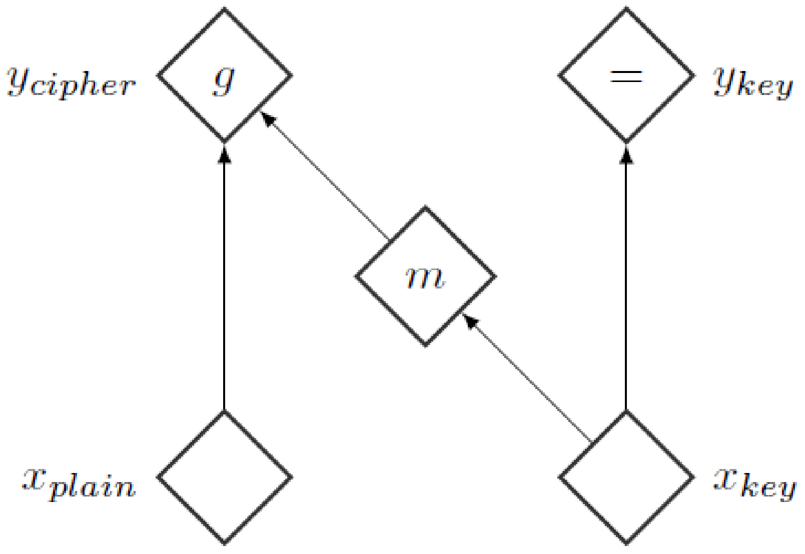


Figure 2: Computational graph of a coupling layer

General coupling layer

Define $y = (y_{I_1}, y_{I_2})$, where:

$$\begin{gathered} y_{I_1} = x_{I_1} \\ y_{I_2} = g(x_{I_2}; m(x_{I_1})) \end{gathered}$$

Coupling raw(耦合规则)

$$g : \mathbb{R}^{(d+1,D)} \times m(\mathbb{R}^{(1,d)}) \rightarrow \mathbb{R}^{(d+1,D)}$$

则耦合层映射的Jacobian determinant：

$$\begin{gathered} \frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_{I_1}}{\partial x_{I_1}} & \frac{\partial y_{I_1}}{\partial x_{I_2}} \\ \frac{\partial y_{I_2}}{\partial x_{I_1}} & \frac{\partial y_{I_2}}{\partial x_{I_2}} \end{bmatrix} = \begin{bmatrix} I_d & 0 \\ \frac{\partial y_{I_2}}{\partial x_{I_1}} & \frac{\partial y_{I_2}}{\partial x_{I_2}} \end{bmatrix}, which\ is\ a\ block\ triangular\ matrix \\ det\frac{\partial y}{\partial x} = det\begin{bmatrix} I_d & 0 \\ \frac{\partial y_{I_2}}{\partial x_{I_1}} & \frac{\partial y_{I_2}}{\partial x_{I_2}} \end{bmatrix} = |I_d||\frac{\partial y_{I_2}}{\partial x_{I_2}}| = |\frac{\partial y_{I_2}}{\partial x_{I_2}}| = det\frac{\partial y_{I_2}}{\partial x_{I_2}} \end{gathered}$$

Inverted mapping is

$$\begin{gathered} x_{I_1} = y_{I_1} \\ x_{I_2} = g^{-1}(y_{I_2}; m(y_{I_1})) = g^{-1}(y_{I_2}; m(x_{I_1})) \end{gathered}$$

Coupling function/transformer m可以是任意实现$d \rightarrow (D-d)$维映射($m : \mathbb{R}^{(1,d)} \rightarrow \mathbb{R}^{(d+1,D)}$)的函数，例如 Deep Neural Network.

Different coupling transformer

Additive Coupling transformer

Defined as $g(a; b) = a + b$, where $a = x_{I_2}$ and $b = m(x_{I_1})$

hence, mapping function and Jacobian matrix are

$$\begin{gathered} y_{I_1} = x_{I_1} \\ y_{I_2} = x_{I_2} + m(x_{I_1}) \\ det\frac{\partial y_{I_2}}{\partial x_{I_2}} = 1 \end{gathered}$$

Inverted mapping is

$$\begin{gathered} x_{I_1} = y_{I_1} \\ x_{I_2} = y_{I_2} - m(y_{I_1}) \end{gathered}$$

Multiplicative coupling transformer

Defined as $g(a;b) = a \circ b$, where $a = x_{I_2}$ and $b = m(x_{I_1}), b \neq 0$

Affine coupling transformer

https://paperswithcode.com/method/affine-coupling

Defined as $g(a;b) = a \circ b_{I_1} + b_{I_2}$, where $a = x_{I_2}$ and $b_{I_1} \neq 0$, $m : \mathbb{R}^{(1,d)} \rightarrow \mathbb{R}^{(d+1,D)} \times \mathbb{R}^{(d+1,D)}$

Non-affine neural transformer

Defined as $g(a_i;b_i) = \omega_{i0} + \sum_{k=1}^{K} \omega_{ik}\sigma(\alpha_{ik}a_i + b_{ik})$

Integration-based transformer

Defined as $g(a_i;b_i) = \int_0^{a_i} h(a;\omega_i)da + b_i$

Neural spline flows

为了克服Integration-based transformer不易求逆，NSF使用K个分段函数作为transformer

Coupling Flows

通过combining multiple coupling layers,搭建coupling flows，实现更复杂的映射。

论文中通过计算Jacobian，证明通过连接三层以上的coupling layer，可以保证每个维度数据都得到变换，通常使用四层。

> 可以Real-NVP中的alternative pattern

Summary

- It is easy to compute inverted mapping and Jacobian determinant（$\mathbb{R}^{(1,d)}$恒等映射是关键，保证逆映射容易求）
- 可以使用任意的变换作为映射函数，例如使用CNN，在高维空间依然适用
- 实现非线性变换

Autoregressive Flows

Basics

基于Coupling flow的思想，改变coupling raw，Coupling function/transformer m is called Conditioner in autoregressive flows.

1. 不对x进行切分，autoregressive flows中，除了x, y，引入另一个input，$m(x_{I_2}) \rightarrow Conditioner(input)$

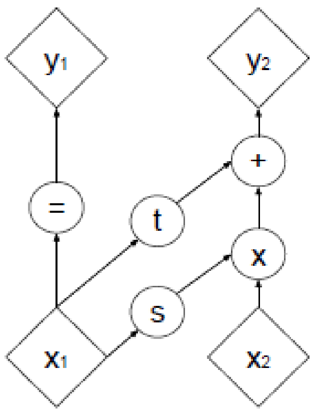
2. 引入时间变量(flow次数)：t

Autoregressive的理论基础与前提：

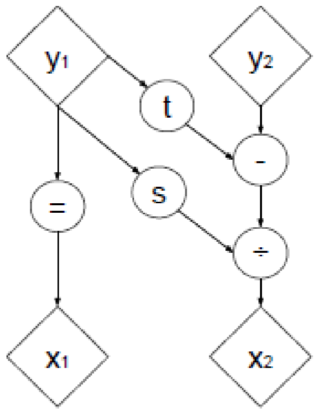
- 三角矩阵的行列式是其对角元素的乘积
- 第i维变量x_i只依赖上一次生成的变量x_{i-1}，即

$$\begin{aligned} p(x_i|x_{1:i-1}) &= N(x_i|\mu_i,(exp\alpha_i)^2) \\ \mu_i &= f_{\mu_i}(x_{1:i-1}) \\ \alpha_i &= f_{\alpha_i}(x_{1:i-1}) \end{aligned}$$

Real-NVP(non-volume preserving)



(a) Forward propagation



(b) Inverse propagation

Refered to affine coupling layer, mapping function and inverse in Real-NVP is defined as

$$\begin{aligned} y_{1:d} &= x_{1:d} \\ y_{d+1:D} &= x_{d+1:D} \circ exp(s(x_{1:d})) + t(x_{1:d}) \qquad (1) \end{aligned}$$

$$\Leftrightarrow$$

$$\begin{aligned} x_{1:d} &= y_{1:d} \\ x_{d+1:D} &= (y_{d+1:D} - t(x_{1:d})) \circ exp(-s(y_{1:d})) \qquad (2) \end{aligned}$$

eg.deduction of (2) from (1):

$$y_{d+1:D} - t(x_{1:d}) = x_{d+1:D} \circ exp(s(x_{1:d})) \to x_{d+1:D} = \frac{y_{d+1:D} - t(x_{1:d})}{exp(s(x_{1:d}))} \to \\ x_{d+1:D} = (y_{d+1:D} - t(x_{1:d})) \circ exp(-s(y_{1:d}))$$

where s and t stand for scale(α) and translation(μ), and are function from $\mathbb{R}^d \to \mathbb{R}^{D-d}$, and $\circ$ is the element-wise product. Computing the inverse is no more complex than the forward propagation.

The Jacobian of this transformation is

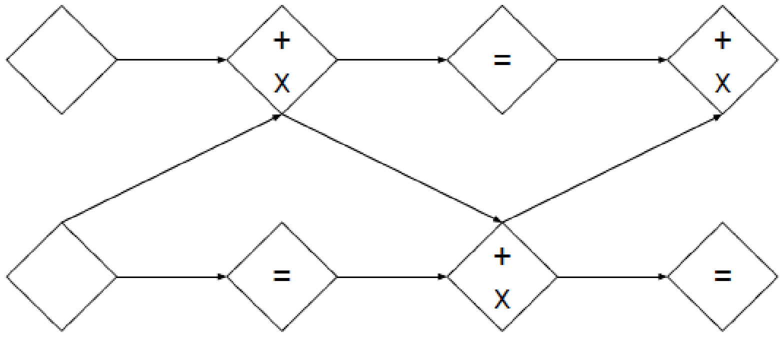
$$\frac{\partial y}{\partial x^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \frac{\partial y_{d+1:D}}{\partial x_{d+1:D}^T} \end{bmatrix} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & diag(exp[s(x_{1:d})]) \end{bmatrix}$$

where $diag(exp[s(x_{1:d})])$ is the diagonal matrix whose diagonal elements correspond to the vector $exp[s(x_{1:d})]$. Hence the Jacobian determinant is $\prod_{i=1}^{d} exp(s(x_i)) = exp^{\sum_{i=1}^{d} s(x_i)}$,which does not involve computing the Jacobian of s or t. Hence s and t can be very complex, eg. deep convolutional neural network. Computing the inverse of the coupling layer does not require computing the inverse of s or t either, so these functions can be arbitrarily complex and difficult to invert.

通常使用Binary mask b 来分割$\mathbb{R}^D$到两个空间，可自定义mask b，例如

- spatial checkerboard pattern mask：it equals 1 while the sum of spatial coordinates is odd, and 0 otherwise.
- channel-wise mask：it equals 1 for the first half of the channel dimensions and 0 for the second half.

Mapping function可简化表示为：$y = b \circ x + (1-b) \circ (x \circ exp(s(b \circ x)) + t(b \circ x))$



(a) In this alternating pattern, units which remain identical in one transformation are modified in the next.

当连接多层coupling layer时:

$$first\ layer : x_b = f_a(x_a)$$
$$second\ layer : y = f_b(x_b) = f_b(f_a(xa)).$$

Inverse and Jacobian determinant依然容易计算，证明如下：

$$\frac{\partial y}{\partial x_a^T} = \frac{f_b(x_b)}{\partial x_a^T} = \frac{f_b(x_b)}{\partial x_b^T} \cdot \frac{x_b}{\partial x_b^T} = \frac{f_b(x_b)}{\partial x_b^T} \cdot \frac{f_a(x_a)}{\partial x_b^T}$$
$$det\frac{\partial y}{\partial x_a^T} = det[\frac{f_b(x_b)}{\partial x_b^T} \cdot \frac{f_a(x_a)}{\partial x_b^T}] = det\frac{f_b(x_b)}{\partial x_b^T} \cdot det\frac{f_a(x_a)}{\partial x_b^T}$$
$$Lemma : (f_b \circ f_a)^{-1} = f_a^{-1} \circ f_b^{-1}$$
$$x_a = f_a^{-1} \circ f_b^{-1}(y)$$

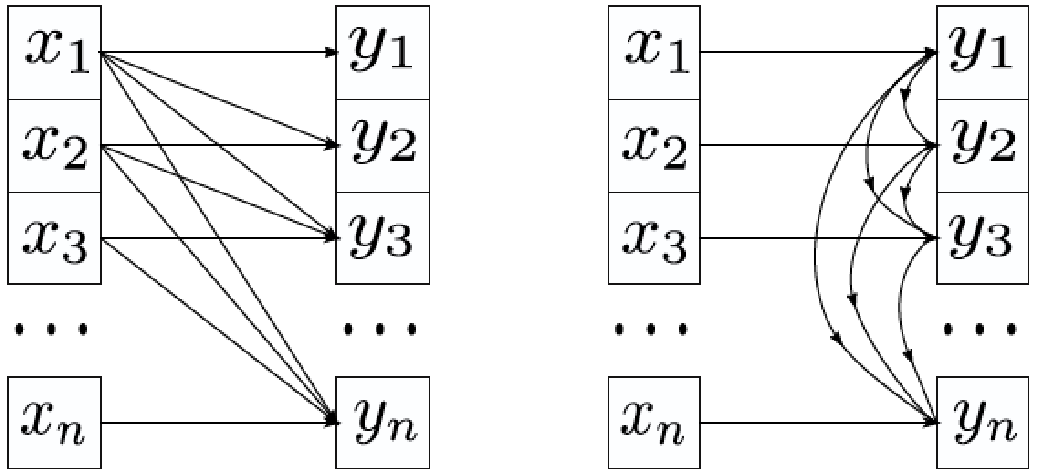
该文章是最早的autoregressive flow探索，应用于图像分类、语音识别等领域。

Masked Autoencoder for Distribution Estimation (MADE)

IAF(Inverse Autoregressive Flow)

类似Real-NVP的autoregressive flow的sampling过程为：首先随机sample一个噪声$\epsilon \sim N(0, I)$,其对应的变量$x_0 = \mu_0 + \sigma_0 \circ \epsilon_0$，当$i > 1$时，$x_i = \mu_i(x_{1:i-1}) + \sigma_i(x_{1:i-1}) \cdot \epsilon_i$，无法并行，sample速度很慢。

IAF优化flow过程，提高采样速度，将上式reparameterization: $x_i = \mu_i(z_{1:i-1}) + \sigma_i(z_{1:i-1}) \cdot z_i$，其中$z \sim q(z)$,即$x_i$不依赖与之前生成的$x_{1:i-1}$，因此可以被**并行采样**。但在inference时，想要得到z_i，还是要根据$z_i = \mu_i + \sigma_i \circ z_{i-1}$,因此**训练慢**。



MAF(masked autoregressive flow)

MAF is mainly pplied to density estimation task, **训练快，采样慢**。

Parallel-Wavenet

同时提高训练和采样速度，是MAF和IAF的结合：用MAF作为teacher model，负责训练是提供分布的指导信息；用IAF作为student model，负责最终的sample。

类似Knowledge distillation的方法，在语音合成上取得了很好的效果，同时也保证了生成语音的速度。

Glow(Generative Flow)

引入可逆的1×1卷积，主要应用于图像生成

WaveGlow and FloWavenet

在语音合成上比较新的方法

Residual Flows

主要应用：分类任务

RevNet

A variant of ResNets where each layer's activations can be reconstructed exactly from the next layer's. Therefore, the activations for most layers need not be stored in memory during backpropagation.

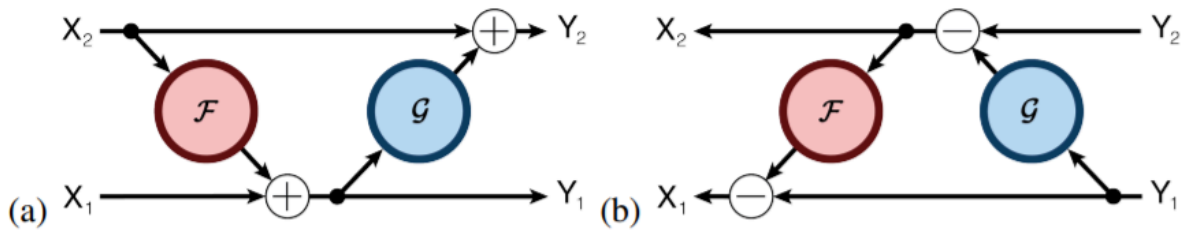


Figure 2: **(a)** the forward, and **(b)** the reverse computations of a residual block, as in Equation 8.

Each reversible block takes input $(x_1 \in \mathbb{R}^d, x_2 \in \mathbb{R}^{D-d})$, output $(y_1 \in \mathbb{R}^d, y_2 \in \mathbb{R}^{D-d})$, and functions $F(.) : \mathbb{R}^d \to \mathbb{R}^{D-d}, G(.) : \mathbb{R}^{D-d} \to \mathbb{R}^d$.

Transformation and inverse:

$$\begin{gathered} y_{1:d} = x_{1:d} + F(x_{d+1:D}) \\ y_{d+1:D} = x_{d+1:D} + G(y_{1:d}) \\ \\ \Leftrightarrow \\ x_{d+1:D} = y_{d+1:D} - G(y_{1:d}) \\ x_{1:d} = y_{1:d} - F(x_{d+1:D}) = y_{1:d} - F(y_{d+1:D} - G(y_{1:d})) \end{gathered}$$

But computation of the Jacobian determinant is inefficient, the Jacobian determinant is not triangular matrix:

$$\begin{bmatrix} \mathbb{I}^d & \frac{\partial y^d}{\partial x^{D-d}} \\ \frac{\partial y^{D-d}}{\partial x^d} & \frac{\partial y^{D-d}}{\partial x^{D-d}} \end{bmatrix}$$

iResNet

Proposition of residual connection: A residual connection is invertible, if the **Lipschitz constant** of the residual block is $Lip(F) < 1$.

- Adding a simple normalization step during training, ensures $Lip(F) < 1$
- Introduce a tractable approximation to the Jacobian log-determinant of a residual block.
- Define a generative model which can be trained by maximum likelihood on unlabeled data.

Infinitesimal Flows

Using Hutchinson's trace estimator to give a scalable unbiased estimate of the log-density. The result is a continuous-time invertible generative model with unbiased density estimation and one-pass sampling, while allowing unrestricted neural network architectures. We demonstrate our approach on **high-dimensional density estimation, image generation, and variational inference**, achieving the state-of-the-art among exact likelihood methods with efficient sampling.

Appendix

KL散度

Kullback-Leibler散度（Kullback-Leibler divergence），又称为相对熵（relative entropy），是两个概率分布（probability distribution）间差异的非对称性度量；在信息理论中，相对熵等价于两个概率分布的信息熵（Shannon entropy）的差值。

相对熵是一些优化算法，例如最大期望算法（Expectation-Maximization algorithm, EM）的损失函数。此时参与计算的一个概率分布为真实分布，另一个为理论（拟合）分布，相对熵表示使用理论分布拟合真实分布时产生的信息损耗 。

熵（entropy）是信息论中重要的信息度量单位，概率分布的熵定义为：

$$H = -\sum_{i-1}^{N} p(x_i) \cdot logp(x_i)$$

相对熵定义为拟合分布对应的熵与实际分布熵的差值，即假设$p(x_i)$为观测到的实际概率分布，$q(x_i)$为拟合的分布用来近似$p(x_i)$，则p, q间的KL散度为：

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \cdot (logp(x_i) - logq(x_i)) = \sum_{i=1}^{N} p(x_i) \cdot log\frac{p(x_i)}{q(x_i)}$$

以期望的形式表示为：

$$D_{KL}(p||q) = E_{p(x)}[logp(x) - logq(x)]$$

- 相对熵不是对称量，即$D_{KL}(p||q) = \sum p \cdot log\frac{p}{q} \neq D_{KL}(q||p) = \sum q \cdot log\frac{q}{p}$
- 相对熵大于等于零，当且仅当$p(x) = q(x)$时等号成立，即$D_{KL}(p||q) \geq 0$，证明如下：

证明：

> 设实直线上的函数$f(x)$是一个非负函数，且$\int_{-\infty}^{\infty} f(x)dx = 1$；$g(.)$是任意的实函数，$\psi$为凸函数，则根据Jensen不等式$\psi(\int_{-\infty}^{\infty} g(x)f(x)dx) \leq \int_{-\infty}^{\infty} \psi(g(x)) \cdot f(x)dx$;
>
> 针对KL散度$D_{KL}(p||q)$,令$\psi(x) = -ln(x) \leftrightarrow$ 凸函数$, f(x) = p(x), g(x) = \frac{q(x)}{p(x)}$，代入Jensen不等式中：

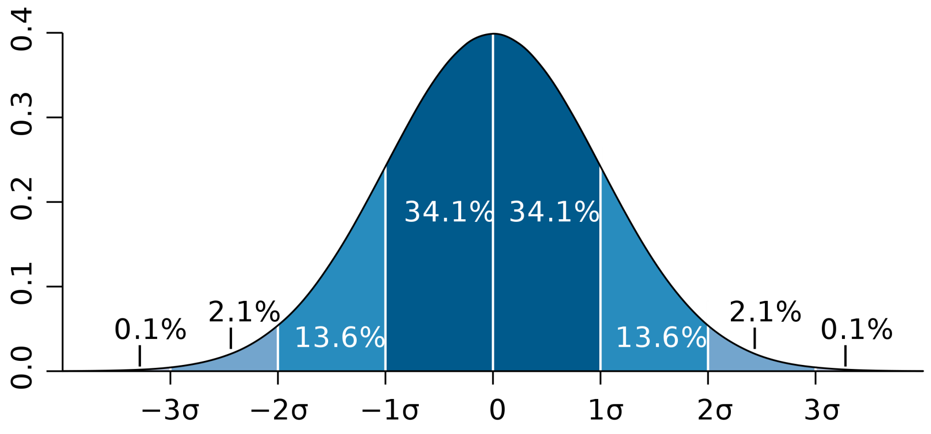
> $$D_{KL}(p||q) = \int p(x) \cdot (-log[\frac{q(x)}{p(x)}])dx \geq -log(\int q(x)dx) = 0$$
>
> 当且仅当$p(x) = q(x)$时，对所有的x，等号才成立。

Probability distribution & Probability density function

In probability theory and statistics, a probability distribution is the mathematical function that gives the probabilities of occurrence of different possible outcomes for an experiment. It is a mathematical description of a random phenomenon in terms of its sample space and the probabilities of events. For instance, if X is used to denote the outcome of a coin toss ("the experiment"), then the probability distribution of X would take the value 0.5 for X = heads, and 0.5 for X = tails (assuming that the coin is fair).

Continuous probability distributions can be described in several ways. The probability density function describes the infinitesimal probability of any given value, and the probability that the outcome lies in a given interval can be computed by integrating the probability density function over that interval.

Probability density function of standard gaussian distribution is as follows:



多维正态分布

假设n维变量$X = (x_1, x_2, ..., x_n)^T$高斯分布，则多维变量的概率分布密度为：

$$p(X) = \frac{1}{2\pi^{\frac{n}{2}}} \cdot \frac{1}{|\sum|^{\frac{1}{2}}} \cdot exp[-\frac{1}{2} \cdot (X-\mu)^T \cdot (\sum)^{-1} \cdot (X-\mu)]$$

其中μ为均值，$\sum$为协方差矩阵：

$$\sum = \begin{bmatrix} cov(x_1,x_1) & cov(x_1,x_2) & ... & cov(x_1,x_n) \\ cov(x_2,x_1) & cov(x_2,x_2) & ... & cov(x_2,x_n) \\ ... & ... & ... & ... \\ cov(x_n,x_1) & cov(x_n,x_2) & ... & cov(x_n,x_n) \end{bmatrix}$$
$$cov(x_i,x_j) = E[(x_i - E(x_i))(x_j - E(x_j))] = E(x_i \cdot x_j) - E(x_i) \cdot E(x_j)$$

当各个变量x_i间相互独立，$cov(x_i,x_j)=\begin{cases} var(x_i) \ \ if \ i=j \\ 0 \ \ otherwise \end{cases}$ ，$\sum = triangular(var(x_i))$；并且当各个变量均服从标准正态分布$N(0,1)$时，$\sum = I_n, |\sum| = 1$，因此概率分布函数可简化为：

$$p(X)=\frac{1}{2\pi^{\frac{n}{2}}}\cdot exp[-\frac{1}{2}\cdot X^T\cdot X]$$
$$ln(p(X))=-\frac{1}{2}X^TX-\frac{n}{2}ln(2\pi)$$

PLU decomposition

pertumation matrix & lower triangular & row echelon form

$$A=\begin{bmatrix}0&1&2&1&2\\1&0&0&0&1\\2&1&2&1&5\\1&2&4&3&6\end{bmatrix}=P_{1\leftrightarrow 2}P_{3\leftrightarrow 4}\begin{bmatrix}1&0&0&0\\0&1&0&0\\1&2&1&0\\2&1&0&1\end{bmatrix}\begin{bmatrix}1&0&0&0&1\\0&1&2&1&2\\0&0&0&1&1\\0&0&0&0&1\end{bmatrix}$$

正定矩阵

正定矩阵判定方法：

- **顺序主子式法**：顺序主子式行列式**都大于零**
  - 顺序主子式：左上角开始，n阶方阵，$A_{11}, A_{22}, \ldots$
  - 行列式都小于零，负定
  - 半正定：偶次大于零（eg. A_{22}），奇次小于零
  - 半负定：偶次小于零，奇次大于零
- 特征值法
- 定义法

eg.

$$f(x,y)=x^2-xy+y^2, \begin{pmatrix}1&-0.5\\-0.5&1\end{pmatrix}, |A_{11}|=1, |A_{22}|=2$$

eg.

已知二次曲线：$L: 9x^2+4y^2+18x+16y-11=0$，矩阵$A=\begin{pmatrix}0.5&0\\0&0.333\end{pmatrix}$，$B=(0.5\ 0.666)^T$，求L在变换$TX=AX+B$下所得方程?

$$AX+B=\begin{pmatrix}0.5&0\\0&0.333\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}+\begin{pmatrix}0.5\\0.666\end{pmatrix}=\begin{pmatrix}0.5x+0.5\\0.333y+0.666\end{pmatrix},$$

令$\begin{cases}m=0.5x+0.5\rightarrow x=2m-1\\ n=0.333y+0.666\rightarrow y=3n-2\end{cases}$，带入曲线方程得$: m^2+n^2=1$，即新方程为$: x^2+y^2=1$

逆函数

x与y表示n维变量，$f(.)$为可逆变换且逆函数为

$$g(.) = f^{-1}(.)$$
$$x = f^{-1}(y) = g(f(x))$$

Jacobian Matrix

设Y=F(X)，F为可逆变换

$$J = \frac{dy}{dx}, \begin{bmatrix} dy1 \\ dy2 \\ dy3 \\ .. \\ dy_{n-1} \\ dy_n \end{bmatrix} = J(\theta_1, \theta_2, ..., \theta_n) \begin{bmatrix} dx1 \\ dx2 \\ dx3 \\ .. \\ dx_{n-1} \\ dx_n \end{bmatrix}$$
$$J^{-1} = \frac{dx}{dy}, \begin{bmatrix} dx1 \\ dx2 \\ dx3 \\ .. \\ dx_{n-1} \\ dx_n \end{bmatrix} = J^{-1}(\theta_1, \theta_2, ..., \theta_n) \begin{bmatrix} dy1 \\ dy2 \\ dy3 \\ .. \\ dy_{n-1} \\ dy_n \end{bmatrix}$$

行列式

性质1：$|AB| = |A||B|$

性质2：$|I| = 1$

性质3：$det(AA^{-1}) = detI = 1$，因此$detA^{-1} = \frac{1}{detA}$

性质4：$det(A \cdot B) = det(A)det(B)$

逆矩阵

对于n阶矩阵A，若有一个n阶矩阵B使得AB=BA=I，则称A可逆，B为A的逆矩阵，记为$A^{-1} = B$

求法：初等行变换$(A|I) \rightarrow (I|A^{-1})$

eg.

$$(A|I) = \begin{pmatrix} 2 & 3 & 3 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 2 & 2 & 1 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 2 & 3 & 3 & 1 & 0 & 0 \\ 2 & 2 & 1 & 0 & 0 & 1 \end{pmatrix} =$$

$$\begin{pmatrix}1 & 1 & 0 & 0 & 1 & 0\\ 0 & 1 & 3 & 1 & -2 & 0\\ 0 & 0 & 1 & 0 & -2 & 1\end{pmatrix} = \begin{pmatrix}1 & 0 & 0 & -1 & -3 & 3\\ 0 & 1 & 0 & 1 & 4 & -3\\ 0 & 0 & 1 & 0 & -2 & 1\end{pmatrix}, A^{-1} = \begin{pmatrix}-1 & -3 & 3\\ 1 & 4 & -3\\ 0 & -2 & 1\end{pmatrix}$$

$$A = \begin{pmatrix}a & b\\ c & d\end{pmatrix}, A^{-1} = \frac{1}{|A|}\begin{pmatrix}d & -b\\ -c & a\end{pmatrix}$$

Lipschitz continuity

在数学中，特别是实分析，利普希茨连续（Lipschitz continuity）以德国数学家鲁道夫·利普希茨命名，是一个比通常连续更强的光滑性条件。直觉上，利普希茨连续函数**限制了函数改变的速度**，符合利普希茨条件的**函数的斜率**，必小于一个称为利普希茨常数（Lipschitz Constant）的实数（该常数依函数而定）。

Likelihood-free inference

On the machine learning side: In machine learning, you usually try to maximize $p(y|x)$, where x is the target, and y is the input (for example, x could be some random noise, and y would be an image). Now, how do we optimize this? A common way to do it, is to assume, that $p(y|x) \sim N(y|\mu(x), \sigma)$. If we assume this, it leads to the **mean squared error**. Note, we *assumed* as form for p(y|x). However, if we don't assume any certain distribution, it is called **likelihood-free learning**.

Why do GANs fall under this? Well, the Loss function is a neural network, and this neural network is not fixed, but learned jointly. Therefore, we dont assume any form anymore (except, that p(y|x)p(y|x) falls in the family of distributions, that can be represented by the discriminator, but for theory sake we say it is a universal function approximator anyway).