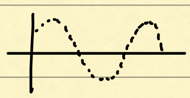
Flow based는 NLL을 이용해, p(x) 확률 분포를 명시적으로 학습한다.

ㄴ Normalization flow 이용

밀도 추정은 기계학습에서 유용히 쓰인다.

- density estimation이란 관측된 데이터들의 분포로부터 원래 분포의 확률 분포 특성을 유추해 내는 것이다.

  예를들어, 어떤 사진이 고양이인 분포가 존재하고 고양이 사진은 그 분포에서 하나의 데이터다.

  예를들어,  에서 sinθ 인걸 추정할 수 있다.

  GAN에서 latent z로부터 x를 생성해 내는 것도 z space에서 x의 확률밀도를 추정하는 것이다. 잘 추정할수록 좋은 generate가 됨

- parametric density estimation

  확률의 분포를 미리 정하고 데이터로부터 parameter만 추정

- non-parametric density estimation

  확률 분포에 대한 사전 지식이 없는데 데이터로만 추정하는 것.

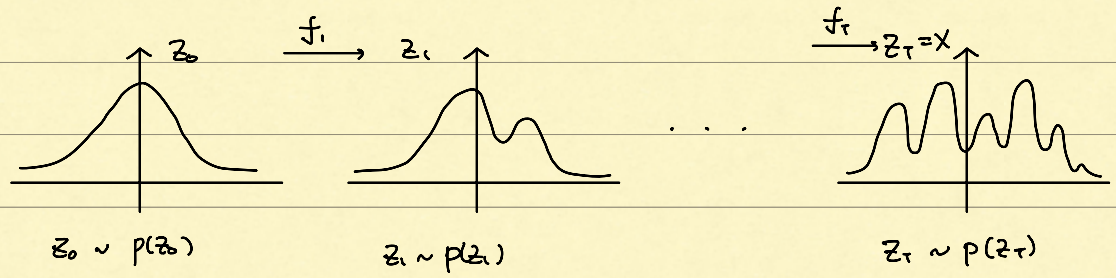
  - kernel density estimation

    kernel은 원점 대칭, 양수, 적분값 1인 함수다. (ex) gaussian, uniform)

    고차원에서는 data가 너무 많이 필요하니, interpolation 하는 것과 비슷

하지만 밀도 측정은 deep learning 에 쉽게 적용하기가 힘들다. 역전파를 수행해야 하기 때문에 보통 쉬운 Gaussian distribution을 latent z에서 가정한다.

Normalizing flow (NF)는 더 좋은 확률 분포 근사를 제시한다.
역변환이 가능한 함수 f로 간단한 분포에 변환을 연속적으로 적용하여,
복잡한 분포를 생성한다.
연속적인 변환의 흐름을 통해, 변수변환에 따른 새로운 변수로 바꾸고,
결국 최종목표 변수 분포를 얻을 수 있다.



- 변수변환 (change of Variable)

Random Variable z가 있을 때, $pdf(z) \sim \pi(z)$

$x = f(z)$ 이고, $pdf(x) \sim p(x)$ 일 때,

$$\int p(x)dx = \int \pi(z)dz = 1$$

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

$$= \pi(f^{-1}(x))\left|{f^{-1}}'(x)\right|$$

multivariable

$$= \pi(f^{-1}(x))\underbrace{\left|\det\frac{df^{-1}}{dx}\right|}_{\text{Jacobian determinant.}}$$

$\Rightarrow \left|\det\frac{df^{-1}}{dx}\right|$ 은 두 변수 x, z의 좌표간 ratio.

$\because\ z_{i-1} \sim p_{i-1}(z_{i-1})\ ,\ z_i = f_i(z_{i-1})\ ,\ z_{i-1} = f^{-1}(z_i)$

$$\Rightarrow p_i(z_i) = p_{i-1}(f_i^{-1}(z_i)) \left| \det \frac{d f_i^{-1}}{d z_i} \right|$$

$$= p_{i-1}(z_{i-1}) \left| \det \left(\frac{d f_i}{d z_{i-1}} \right)^{-1} \right|$$

$$= p_{i-1}(z_{i-1}) \left| \det \frac{d f_i}{d z_{i-1}} \right|^{-1}$$

$$\Rightarrow \log p_i(z_i) = \log p_{i-1}(z_{i-1}) - \log \left| \det \frac{d f_i}{d z_{i-1}} \right|$$

$$\therefore \log p(x) = \log \pi(z_T) = \log \pi_{T-1}(z_{T-1}) - \log \left| \det \frac{d f_T}{d z_{T-1}} \right|$$

$$= \log \pi_{T-2}(z_{T-2}) - \log \left| \det \frac{d f_{T-1}}{d z_{T-2}} \right| - \log \left| \det \frac{d f_T}{d z_{T-1}} \right|$$

$$\vdots$$

$$= \log \pi_0(z_0) - \sum_{i=1}^{T} \log \left| \det \frac{f_i}{d z_{i-1}} \right|$$

f : flow

π : normalizing flow

$\Rightarrow$ f는 invertible하고

Jacobian det는

계산하기 쉬워야 좋다.

* Models

Input에 대한 log-likelihood인 logP(x)는 tractable 이기 때문에

flow-based에서 training criterion은 보통 NLL 이다.

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

- RealNVP

RealNVP는 역변환이 가능한 1대1 변환 함수 시퀀스를 적용하여

NF를 수행했다.

각 함수 $f: x \to y$ 는 affine coupling layer 라고 불리며

입력은 두 파트로 나뉜다

$$y_{1:d} = x_{1:d}$$

$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d})$$ (s: scale, t: translation, mapping $\mathbb{R}^d \to \mathbb{R}^{D-d}$)

Condition 1: "It is easily invertible."

Yes and it is fairly straightforward.

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

Condition 2: "Its Jacobian determinant is easy to compute."

Yes. It is not hard to get the Jacobian matrix and determinant of this transformation. The Jacobian is a lower triangular matrix.

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

Hence the determinant is simply the product of terms on the diagonal.

$$\det(\mathbf{J}) = \prod_{j=1}^{D-d} \exp(s(\mathbf{x}_{1:d}))_j = \exp(\sum_{j=1}^{D-d} s(\mathbf{x}_{1:d})_j)$$

→ s, t의 역함수를 계산할 필요가 없고, J 에서도 s, t의 J를 계산할 필요가 없기 때문에 DNN으로 구성할 수 있다.

→ 모든 channel이 대체될수 있도록 계층마다 channel 순서를 뒤집는다.

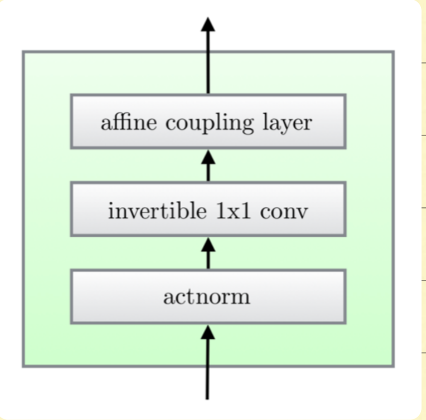
- NICE

RealNVP 에서 scale term이 없어진 것.

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

- Glow

RealNVP의 channel reverse를 1x1 conv로 바꿨다.



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

* Autoregressive flow

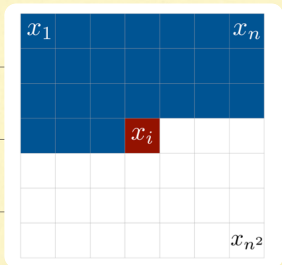
Autoregressive constraint: output이 과거의 시퀀스에 의해 결정되는 것

$$p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i | x_1, \ldots, x_{i-1}) = \prod_{i=1}^{D} p(x_i | x_{1:i-1})$$

=> NF에서 각 vector dimension의 조건이 이전 차원에 따라 결정됨

- PixelRNN

왼쪽 위부터 오른쪽 아래까지 새로운 픽셀을 생성해 내는 generative model 이다.



=> 과거의 확률 분포 조건으로부터 x_i 생성

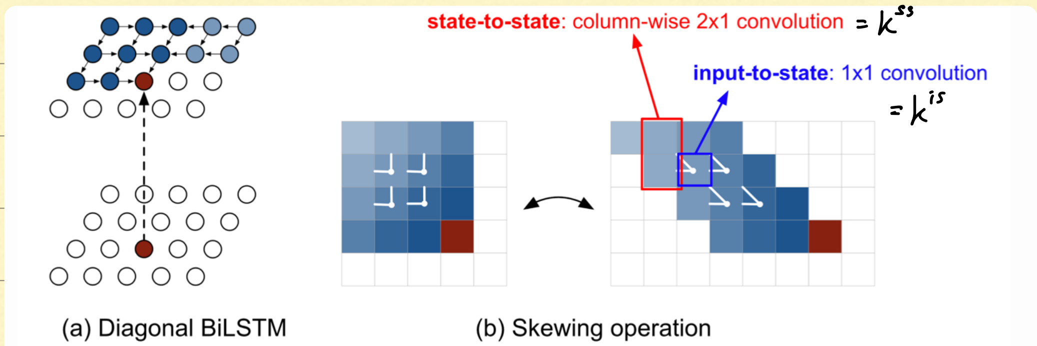


Fig. 7. (a) PixelRNN with diagonal BiLSTM. (b) Skewing operation that offsets each row in the feature map by one with regards to the row above. (Image source: Oord et al, 2016)

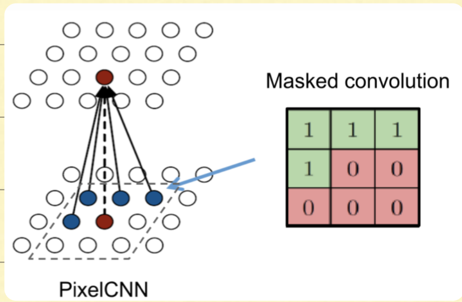
$$[\mathbf{o}_i, \mathbf{f}_i, \mathbf{i}_i, \mathbf{g}_i] = \sigma(\mathbf{K}^{ss} \circledast \mathbf{h}_{i-1} + \mathbf{K}^{is} \circledast \mathbf{x}_i)$$

; σ is tanh for g, but otherwise sigmoid; $\circledast$ is convolution operation.

$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \mathbf{g}_i$$

; $\odot$ is elementwise product.

$$\mathbf{h}_i = \mathbf{o}_i \odot \tanh(\mathbf{c}_i)$$



- WaveNet

pixelRNN이 1-D에서 이루어지는 것.

causal convolution의 stack으로 구성된다. 특정 timestep에서의 output은 과거의 관측 data에만 의존한다.

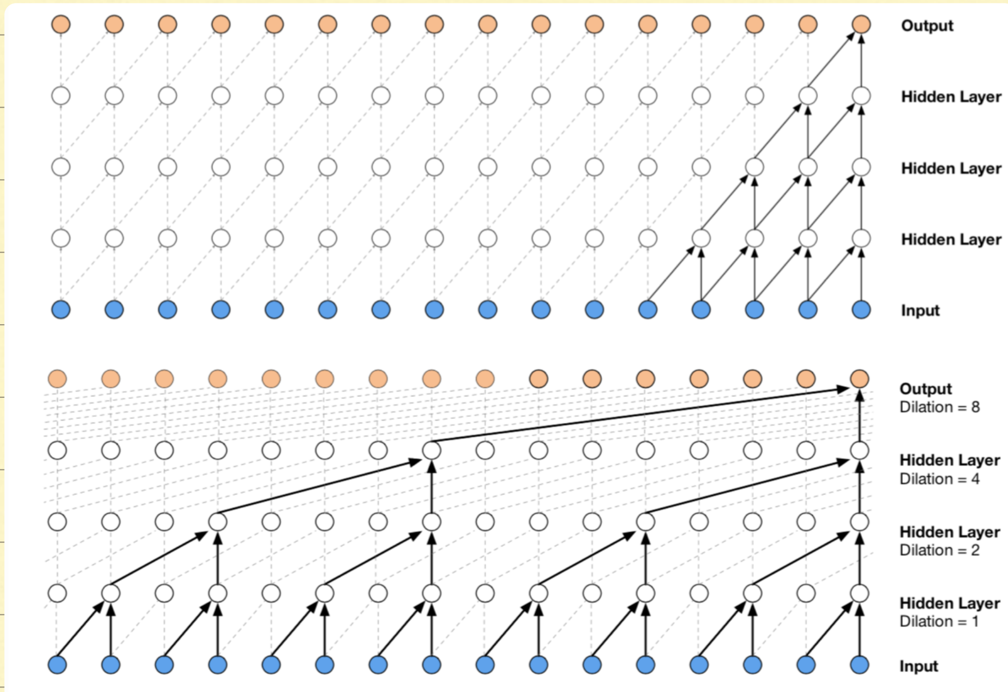


Fig. 9. Visualization of WaveNet models with a stack of (top) causal convolution layers and (bottom) dilated convolution layers. (Image source: Van Den Oord, et al. 2016)

→ 더 긴 시퀀스를 위해 dilated convolution 사용

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

→ residual도 사용

— Masked Autoregressive flow (MAF)

$z \sim \pi(z)$, $x \sim p(x)$ 이고, $\pi(z)$를 알고 있다면.

MAF에서 x_i는 $x_{1:i-1}$의 조건에서부터 생성된다.

- Data generation, producing a new $\mathbf{x}$:

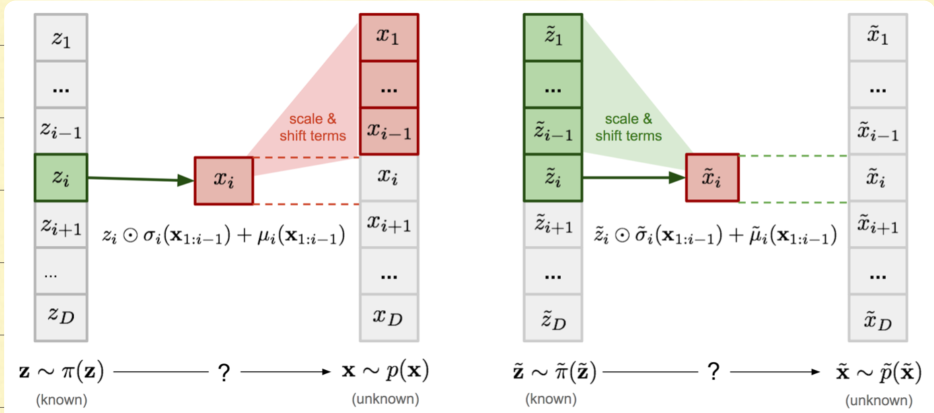
$$x_i \sim p(x_i|\mathbf{x}_{1:i-1}) = z_i \odot \sigma_i(\mathbf{x}_{1:i-1}) + \mu_i(\mathbf{x}_{1:i-1}), \text{ where } \mathbf{z} \sim \pi(\mathbf{z})$$

- Density estimation, given a known $\mathbf{x}$:

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

— Inverse Autoregressive Flow (IAF)

MAF와 같지만 inverse로 이루어진다.



Masked Autoregressive Flow (MAF) Inverse Autoregressive Flow (IAF)

	Base distribution	Target distribution	Model	Data generation	Density estimation
MAF	$\mathbf{z} \sim \pi(\mathbf{z})$	$\mathbf{x} \sim p(\mathbf{x})$	$x_i = z_i \odot \sigma_i(\mathbf{x}_{1:i-1}) + \mu_i(\mathbf{x}_{1:i-1})$	Sequential; slow	One pass; fast
IAF	$\tilde{\mathbf{z}} \sim \pi(\tilde{\mathbf{z}})$	$\tilde{\mathbf{x}} \sim p(\tilde{\mathbf{x}})$	$\tilde{x}_i = \tilde{z}_i \odot \tilde{\sigma}_i(\tilde{\mathbf{z}}_{1:i-1}) + \tilde{\mu}_i(\tilde{\mathbf{z}}_{1:i-1})$	One pass; fast	Sequential; slow