NF는 latent에 대한 posterior을 flexible을 제공한다.

새로운 NF인 IAF 제안 → high-dim으로 확장 잘됨.

autoregressive 기반 invertible transform chain으로 구성.

* Introduction.

Density estim에 사용되는 Gaussian autoregressive 기반.

지정된 순서가 있는 변수 사용. 이전 element를 input으로, 평균, 표준 편차 output.

이전 논문들과는 다르게, 시공간적 변환에 잘 맞고, multiple hierarchical latent로 향상된 성능

* Variational Inference and Learning.

multiple latent에서 보통 inference는 순서가 있는 partial Inference가 된다.

ex) $q(z_a, z_b | x) = q(z_a | x) q(z_b | z_a, x)$

* Requirements for Computational Tractability

Inference model을 효율적으로 bound 하는 것에는

1) $q(z|x)$를 계산하고 미분하는데 효율적.

2) sampling이 효율적.

z가 high-dim일 때, 병렬하는 데에 도움이 됨

* Normalizing flow.

유연한 posterior를 위한 방법 → Jacobian을 알아야함. → limit가 있음.

$$\mathbf{z}_0 \sim q(\mathbf{z}_0|\mathbf{x}), \quad \mathbf{z}_t = \mathbf{f}_t(\mathbf{z}_{t-1}, \mathbf{x}) \quad \forall t = 1...T$$

$$\log q(\mathbf{z}_T|\mathbf{x}) = \log q(\mathbf{z}_0|\mathbf{x}) - \sum_{t=1}^{T} \log \det \left| \frac{d\mathbf{z}_t}{d\mathbf{z}_{t-1}} \right|$$

$$\mathbf{f}_t(\mathbf{z}_{t-1}) = \mathbf{z}_{t-1} + \mathbf{u} h(\mathbf{w}^T \mathbf{z}_{t-1} + b)$$

* Inverse Autoregressive Transformations.

High-dim 에 대한 normalizing flow를 위해, MADE 나 PixelCNN 같은 autoregressive AE를 고려.

y는 $y = \{y_i\}_{i=1}^{D}$, $[\mu(y), \sigma(y)]$

Autoregressive 이기 때문에 $\partial[\mu_i, \sigma_i]/\partial y_j = [0, 0]$ for $j \geq i$

Variational inference는 posterior를 필요로 하니까, IAF는 normalizing flow에 적합

$$\epsilon_i = \frac{y_i - \mu_i(y_{1:i-1})}{\sigma_i(y_{1:i-1})}$$

병렬화 : $\epsilon = (y - \mu(y))/\sigma(y) \longrightarrow$ elemental-wise

IAF는 단순한 Jacobian matrix : $\partial[\mu_i, \sigma_i]/\partial y_j = [0, 0]$ for $j \geq i$

$$\partial \epsilon_i/\partial y_j = 0 \text{ for } j > i \text{ (lower triangular Jacobian)}$$

$$\partial \epsilon_i/\partial y_i = \sigma_i$$

$$\therefore \log \det \left| \frac{d\epsilon}{dy} \right| = \sum_{i=1}^{D} -\log \sigma_i(y)$$

⇒ IAF는 model flexible, parallelizability across dimension, simple log-determinant는 normalize flow에 적합

Algorithm 1: Pseudo-code of an approximate posterior with Inverse Autoregressive Flow (IAF)

Data:

  $\mathbf{x}$: a datapoint, and optionally other conditioning information
  $\boldsymbol{\theta}$: neural network parameters
  $\texttt{EncoderNN}(\mathbf{x}; \boldsymbol{\theta})$: encoder neural network, with additional output $\mathbf{h}$
  $\texttt{AutoregressiveNN}[*](\mathbf{z}, \mathbf{h}; \boldsymbol{\theta})$: autoregressive neural networks, with additional input $\mathbf{h}$
  $\texttt{sum}(.)$: sum over vector elements
  $\texttt{sigmoid}(.)$: element-wise sigmoid function

Result:

  $\mathbf{z}$: a random sample from $q(\mathbf{z}|\mathbf{x})$, the approximate posterior distribution
  l: the scalar value of $\log q(\mathbf{z}|\mathbf{x})$, evaluated at sample 'z'

$[\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{h}] \leftarrow \texttt{EncoderNN}(\mathbf{x}; \boldsymbol{\theta})$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
$\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}$
$l \leftarrow -\texttt{sum}(\log \boldsymbol{\sigma} + \frac{1}{2}\boldsymbol{\epsilon}^2 + \frac{1}{2}\log(2\pi))$
for $t \leftarrow 1$ **to** T **do**
  $[\mathbf{m}, \mathbf{s}] \leftarrow \texttt{AutoregressiveNN}[t](\mathbf{z}, \mathbf{h}; \boldsymbol{\theta})$
  $\boldsymbol{\sigma} \leftarrow \texttt{sigmoid}(\mathbf{s})$
  $\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \mathbf{z} + (1 - \boldsymbol{\sigma}) \odot \mathbf{m}$
  $l \leftarrow l - \texttt{sum}(\log \boldsymbol{\sigma})$
end

* Inverse Autoregressive Flow (IAF)

$\epsilon = (\mathbf{y} - \boldsymbol{\mu}(\mathbf{y}))/\boldsymbol{\sigma}(\mathbf{y})$ 를 base로 함.

$$\log q(\mathbf{z}_T|\mathbf{x}) = \log q(\mathbf{z}_0|\mathbf{x}) - \sum_{t=1}^{T} \log \det \left| \frac{d\mathbf{z}_t}{d\mathbf{z}_{t-1}} \right|$$ 계산.

ㄴ Algorithm 1.

Encoder는 μ_0, σ_0 그리고 flow에 추가 입력이 되는 h를 output으로 한다.

초기 sample : $\mathbf{z}_0 = \boldsymbol{\mu}_0 + \boldsymbol{\sigma}_0 \odot \boldsymbol{\epsilon}$

t : $\mathbf{z}_t = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \odot \mathbf{z}_{t-1}$

$\frac{d\mu_t}{dz_{t-1}}, \frac{d\sigma_t}{dz_{t-1}}$ = triangular with zero diagonal

$\frac{dz_t}{dz_{t-1}}$ = triangular with σ_t diagonal

$$\log q(\mathbf{z}_T|\mathbf{x}) = -\sum_{i=1}^{D} \left(\tfrac{1}{2}\epsilon_i^2 + \tfrac{1}{2}\log(2\pi) + \sum_{t=0}^{T} \log \sigma_{t,i} \right)$$

posterior를 true에 fitting하는 ability는 autoregressive depth에 따라 비례

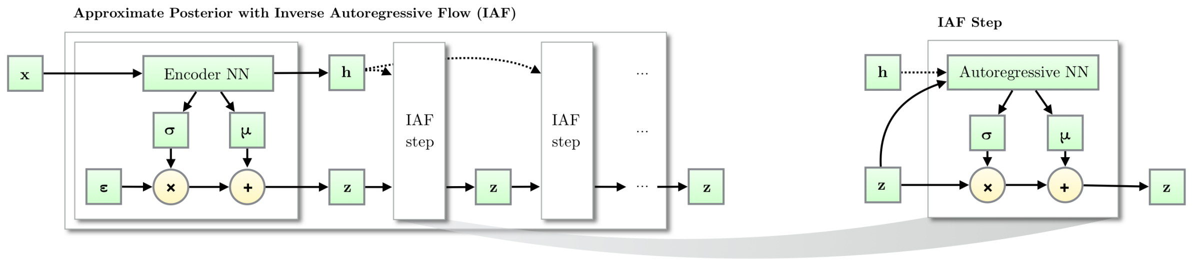

Figure 2: Like other normalizing flows, drawing samples from an approximate posterior with Inverse Autoregressive Flow (IAF) consists of an initial sample **z** drawn from a simple distribution, such as a Gaussian with diagonal covariance, followed by a chain of nonlinear invertible transformations of **z**, each with a simple Jacobian determinants.

Stable version은 unconstraint two vector output $[m_t, s_t]$

$$[\mathbf{m}_t, \mathbf{s}_t] \leftarrow \texttt{AutoregressiveNN}[t](\mathbf{z}_t, \mathbf{h}; \boldsymbol{\theta}) \tag{12}$$

and compute $\mathbf{z}_t$ as:

$$\boldsymbol{\sigma}_t = \text{sigmoid}(\mathbf{s}_t) \tag{13}$$

$$\mathbf{z}_t = \boldsymbol{\sigma}_t \odot \mathbf{z}_{t-1} + (1 - \boldsymbol{\sigma}_t) \odot \mathbf{m}_t \tag{14}$$

Algorithm 1: Pseudo-code of an approximate posterior with Inverse Autoregressive Flow (IAF)

Data:
- $\mathbf{x}$: a datapoint, and optionally other conditioning information
- $\boldsymbol{\theta}$: neural network parameters
- $\texttt{EncoderNN}(\mathbf{x}; \boldsymbol{\theta})$: encoder neural network, with additional output $\mathbf{h}$
- $\texttt{AutoregressiveNN}[*](\mathbf{z}, \mathbf{h}; \boldsymbol{\theta})$: autoregressive neural networks, with additional input $\mathbf{h}$
- $\texttt{sum}(.)$: sum over vector elements
- $\texttt{sigmoid}(.)$: element-wise sigmoid function

Result:
- $\mathbf{z}$: a random sample from $q(\mathbf{z}|\mathbf{x})$, the approximate posterior distribution
- l: the scalar value of $\log q(\mathbf{z}|\mathbf{x})$, evaluated at sample 'z'

$[\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{h}] \leftarrow \texttt{EncoderNN}(\mathbf{x}; \boldsymbol{\theta})$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
$\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} + \boldsymbol{\mu}$
$l \leftarrow -\texttt{sum}(\log \boldsymbol{\sigma} + \frac{1}{2}\boldsymbol{\epsilon}^2 + \frac{1}{2}\log(2\pi))$
for $t \leftarrow 1$ **to** T **do**
  $[\mathbf{m}, \mathbf{s}] \leftarrow \texttt{AutoregressiveNN}[t](\mathbf{z}, \mathbf{h}; \boldsymbol{\theta})$
  $\boldsymbol{\sigma} \leftarrow \texttt{sigmoid}(\mathbf{s})$
  $\mathbf{z} \leftarrow \boldsymbol{\sigma} \odot \mathbf{z} + (1 - \boldsymbol{\sigma}) \odot \mathbf{m}$
  $l \leftarrow l - \texttt{sum}(\log \boldsymbol{\sigma})$
end

---

s_t는 1,2쯤이 양수가 되는 것이 좋다 → z를 조금만 update 해도 됨

Autoregressive Network는 IAF의 rich non-linear transform 형성.

* Results

Table 1: Generative modeling results on the dynamically sampled binarized MNIST version used in previous publications (Burda et al., 2015). Shown are averages; the number between brackets are standard deviations across 5 optimization runs. The right column shows an importance sampled estimate of the marginal likelihood for each model with 128 samples. Best previous results are reproduced in the first segment: [1]: (Salimans et al., 2014) [2]: (Burda et al., 2015) [3]: (Kaae Sønderby et al., 2016) [4]: (Tran et al., 2015)

Model	VLB	$\log p(\mathbf{x}) \approx$
Convolutional VAE + HVI [1]	-83.49	-81.94
DLGM 2hl + IWAE [2]		-82.90
LVAE [3]		-81.74
DRAW + VGP [4]	-79.88	
Diagonal covariance	-84.08 ($\pm$ 0.10)	-81.08 ($\pm$ 0.08)
IAF (Depth $=2$, Width $=320$)	-82.02 ($\pm$ 0.08)	-79.77 ($\pm$ 0.06)
IAF (Depth $=2$, Width $=1920$)	-81.17 ($\pm$ 0.08)	-79.30 ($\pm$ 0.08)
IAF (Depth $=4$, Width $=1920$)	-80.93 ($\pm$ 0.09)	-79.17 ($\pm$ 0.08)
IAF (Depth $=8$, Width $=1920$)	-80.80 ($\pm$ 0.07)	**-79.10** ($\pm$ 0.07)

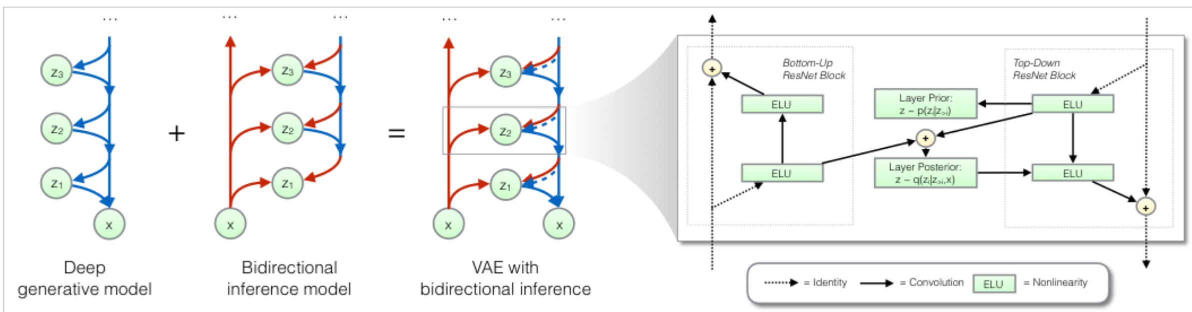

Figure 3: Overview of our ResNet VAE with bidirectional inference. The posterior of each layer is parameterized by its own IAF.