Neural Processes

NTTコミュニケーションズ株式会社
イノベーションセンターテクノロジー部門
藤原大悟

紹介論文

Neural Process (Family)

ほぼ同内容の二本の論文
・Conditional Neural Process (Garnelo et.al., ICML2018)
・Neural Processes (Garnelo et.al., ICML2018WS)

上記の発展形の論文
・Attentive Neural Processes (Kim et.al., ICLR2019)

今回は提唱論文となる上2つを紹介します。

今回のモチベーション：**適応的な(動的な)学習手法について調べたい**

手法の説明

ガウス過程(GP)

Targets $\boldsymbol{f}^* = (f_1^*, f_2^*, \dots, f_M^*)$ for input $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \dots, \boldsymbol{x}_M^*)$
Observation $\mathbf{y} = (y_1, y_2, \dots, y_N)$ for input $(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N)$

特徴：
- 確率過程モデル(関数そのものの確率分布を与えるモデル)
  - 分散の計算なども可能
- 実際の計算では、
  - N個の観測で条件付け
  - M個の予測値の同時分布を求める
- 学習と推論を分離できない
  →条件付確率計算で同時に行う
- 事前学習は分離して行える
  →カーネル関数のハイパラθの学習
  →これはN個の観測データで行う
- 推論が重い($\mathcal{O}(N^3)$)

$$K = \begin{array}{cc} & \begin{array}{cc} N & M \end{array} \\ \begin{array}{c} N \\ M \end{array} & \begin{pmatrix} K_{NN} & K^{NM} \\ K_{NM}^T & K_{MM} \end{pmatrix} \end{array}$$

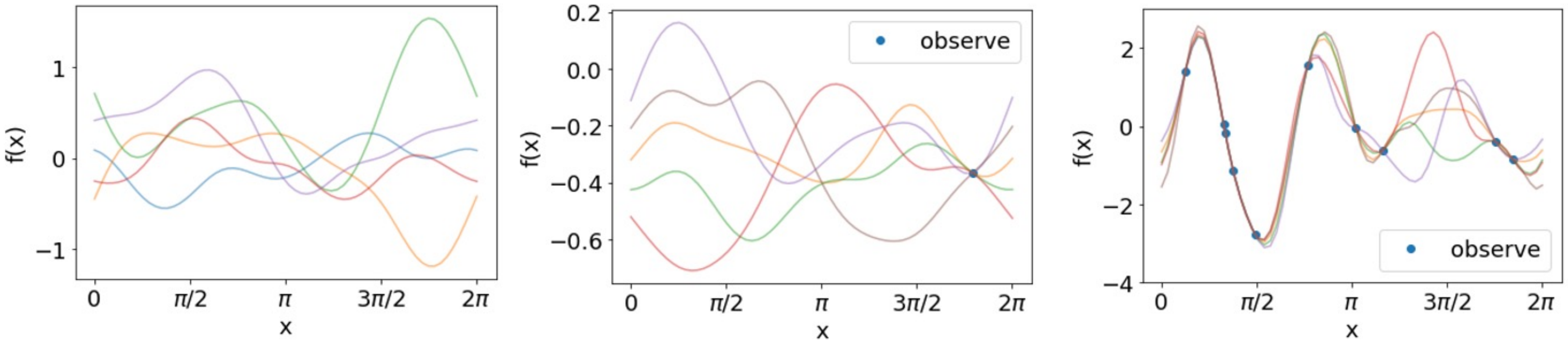
given joint model : $p(\boldsymbol{y}, \boldsymbol{f}^*) \sim \mathcal{N}(\mathbf{0}, K)$

$$\boxed{p(\boldsymbol{f}^* \mid \boldsymbol{y}) \sim \mathcal{N}\left(K_{NM}^T K_{NN}^{-1} \boldsymbol{y}, K_{MM} - K_{NM}^T K_{NN}^{-1} K_{NM}\right)}$$

ここで、

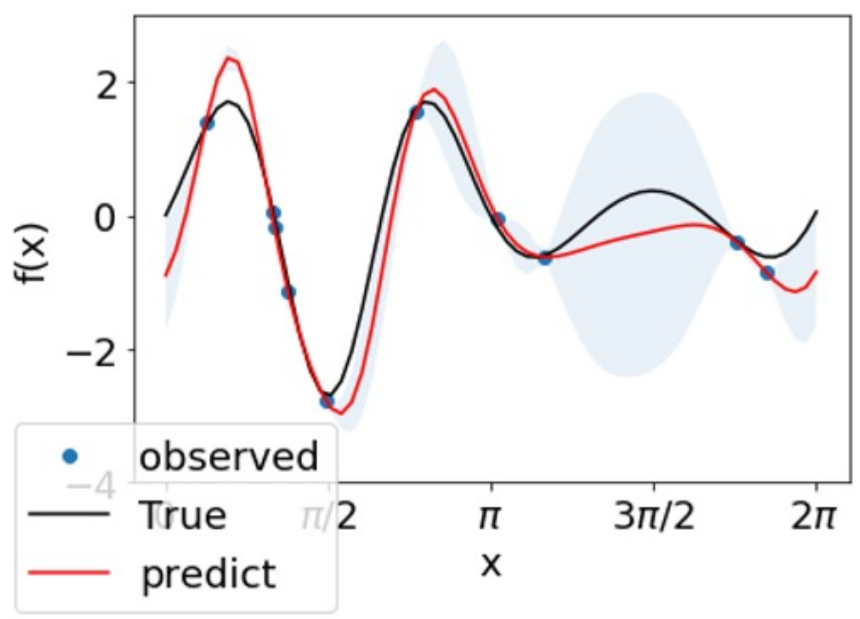
$$K_{NN} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}, \; k(\boldsymbol{x}_i, \boldsymbol{x}_j) = k_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j)$$ （適当なカーネル関数）

observation conditioned sampling :



No data　1 data　10 data

prediction
(mean and variance)



手法説明

コアアイデア：
- **NNの表現能力**を利用しながら
- ガウス過程のような**確率過程モデル**で
- **適応的(動的)な学習**を実現できないか？

再掲：GPの特徴

特徴：
- 確率過程モデル(関数そのものの確率分布を与えるモデル)
  - 分散の計算なども可能
- 実際の計算では、
  - N個の観測で条件付け
  - M個の予測値の同時分布を求める
- 学習と推論を分離できない
  →条件付確率計算で同時に行う
- 事前学習は分離して行える
  →カーネル関数のハイパラθの学習
  →これはN個の観測データで行う
- 推論が重い($\mathcal{O}(N^3)$)

Gaussian Process vs. Neural Network

	Gaussian Process	Neural Network	Neural Process
to black box function	○	○	○
probabilistic	○	△(limited model)	○
flexibility (to new data)	○	×	○
computation cost (in inference)	×	○	○
representation ability	○	◎	◎

手法説明

Neural Processの特徴

特徴：
・確率過程モデル(関数そのものの確率分布を与えるモデル)
　　・分散の計算なども可能
・実際の計算では、
　　・N個の観測で条件付け
　　・M個の予測値の同時分布を求める
・学習と推論を分離できない
　→条件付確率計算で同時に行う
・事前学習は分離して行える
　→カーネル関数のハイパラθの学習
　→これはN個の観測データで行う
~~・推論が重い($O(N^3)$)~~

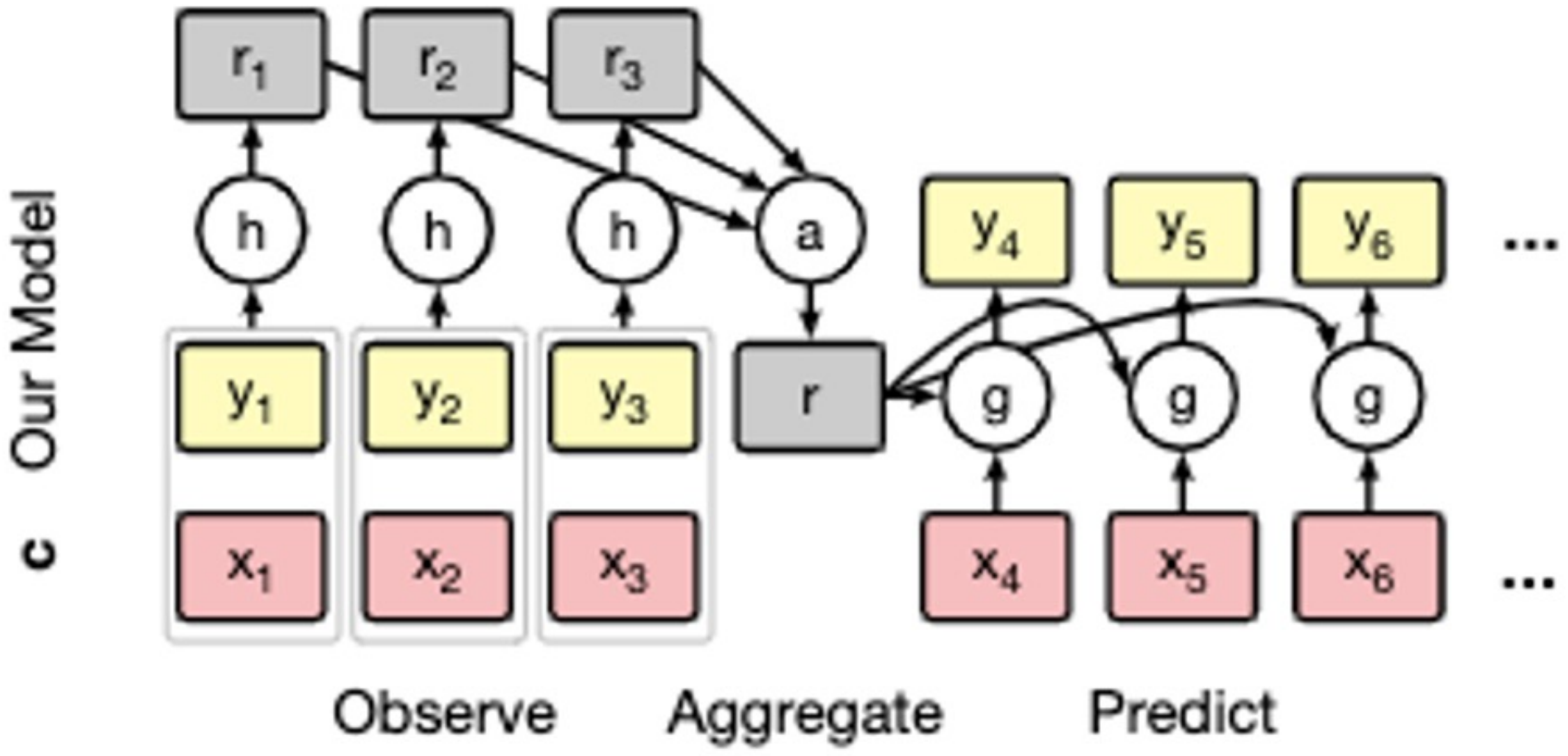
コアアイデア：
- **NNの表現能力**を利用しながら
- ガウス過程のような**確率過程モデル**で
- **適応的(動的)な学習**を実現できないか？

Gaussian Process vs. Neural Network

	Gaussian Process	Neural Network	Neural Process
to black box function	◯	◯	◯
probabilistic	◯	△(limited model)	◯
flexibility (to new data)	◯	×	◯
computation cost (in inference)	×	◯	◯
representation ability	◯	◎	◎

Conditional Neural Process

- architecture : ⟶
- formulation :



observations $O = \{(x_i, y_i)\}_{i=1}^{N} \subset X \times Y$

targets $T = \{x_i\}_{i=N+1}^{N+M} \subset X$

$$r_i = h_\theta(x_i, y_i) \quad \forall (x_i, y_i) \in O$$

representation vector : $r = r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n$

distribution parameter : $\phi_i = g_\psi(x_i, r) \quad \forall (x_i) \in T$

$$Q_{\theta\psi}(f(x_i) \mid O, x_i) = Q(f(x_i) \mid \phi_i)$$

e.g. $\phi_i = (\mu_i, \sigma_i^2)$ in $\mathcal{N}(\mu_i, \sigma_i^2)$

h_θ, g_ψ : Neural Networks

$a(r_{1:N}) = \oplus$: commutative operation

e.g. $r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n = \frac{1}{n}\sum_{i=1}^{n} r_i$

- Points:

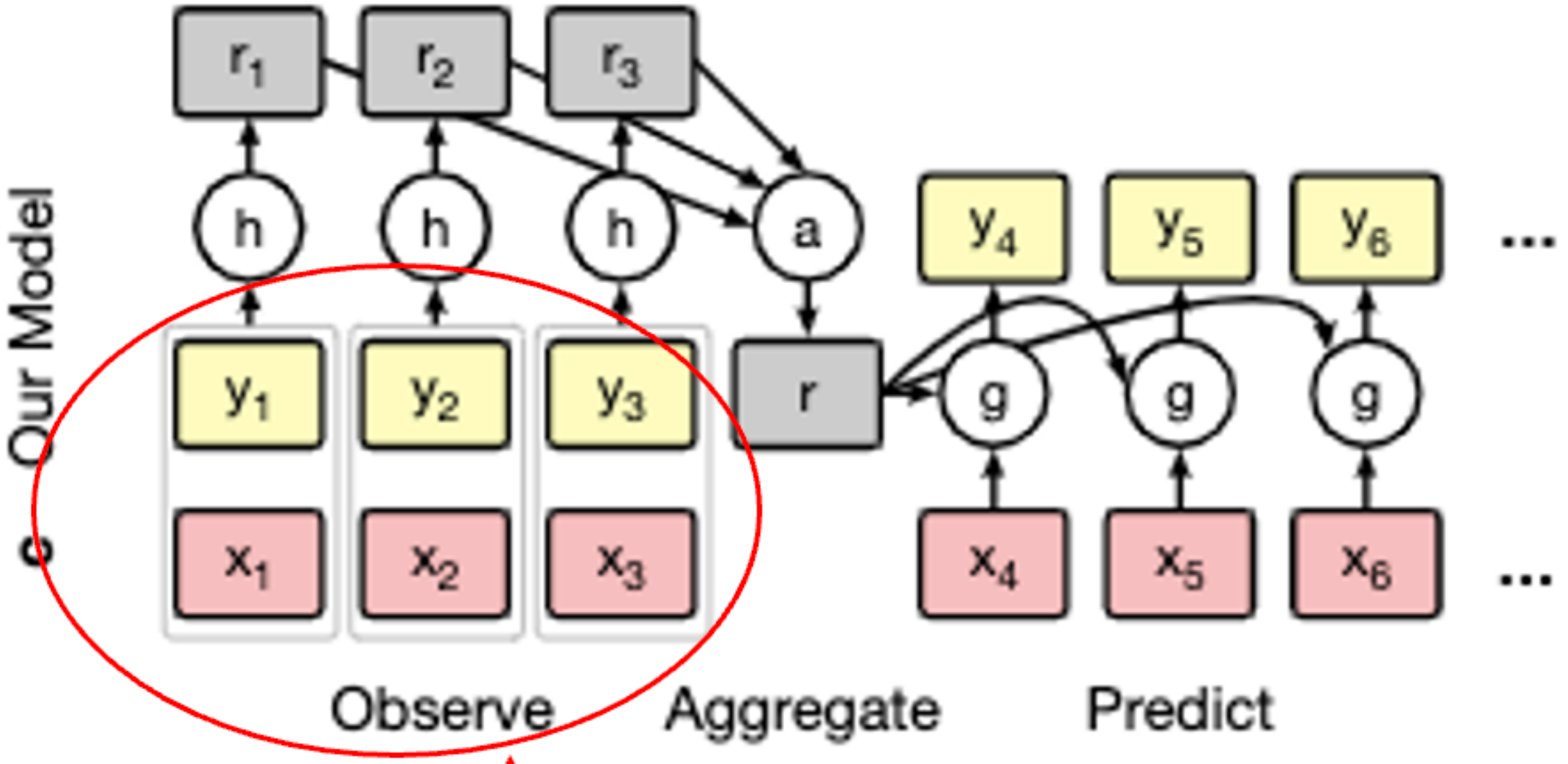
ざっくりいうとどういうモデル？？

θ, ψの事前学習を行なった後、
(Lossなどは後述)
推論そのものはN個のデータで条件づけて行う
→学習の逐次実行
→適応性につながる

学習器の学習(Meta-Learning)

Conditional Neural Process

- architecture :
- formulation :



observations $O = \{(x_i, y_i)\}_{i=1}^{N} \subset X \times Y$

targets $T = \{x_i\}_{i=N+1}^{N+M} \subset X$

$$r_i = h_\theta (x_i, y_i) \quad \forall (x_i, y_i) \in O$$

representation vector : $r = r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n$

distribution parameter : $\phi_i = g_\psi (x_i, r) \quad \forall (x_i) \in T$

$$Q_{\theta\psi} (f (x_i) \mid O, x_i) = Q (f (x_i) \mid \phi_i)$$

e.g. $\phi_i = (\mu_i, \sigma_i^2)$ in $\mathcal{N} (\mu_i, \sigma_i^2)$

h_θ, g_ψ : Neural Networks

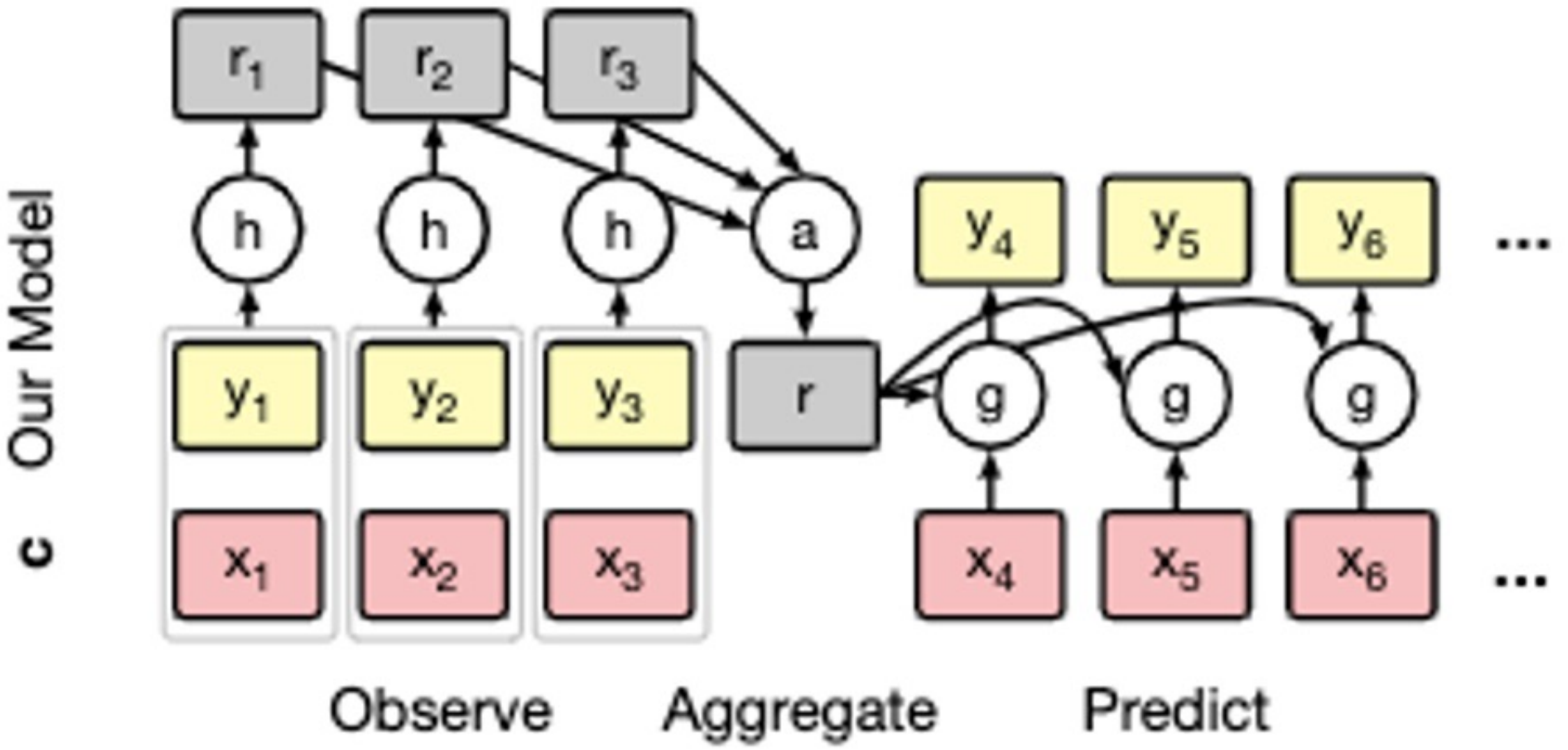
$a(r_{1:N}) = \oplus$: commutative operation

e.g. $r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n = \frac{1}{n} \sum_{i=1}^{n} r_i$

- Points:
- **Reuse training data also in test (prediction) phase**
  - thought as shifting training cost from training phase to test phase
- **Scalability**
  - in test phase, this model scales O($N + M$), otherwise GP scales O(N^3)

Conditional Neural Process

- architecture :
- formulation :



observations $O = \{(x_i, y_i)\}_{i=1}^{N} \subset X \times Y$

targets $T = \{x_i\}_{i=N+1}^{N+M} \subset X$

$$r_i = h_\theta(x_i, y_i) \quad \forall (x_i, y_i) \in O$$

representation vector : $r = r_1 \oplus r_2 \oplus \dots r_{n-1} \oplus r_n$

distribution parameter : $\phi_i = g_\psi(x_i, r) \quad \forall (x_i) \in T$

$$Q_{\theta\psi}(f(x_i) \mid O, x_i) = Q(f(x_i) \mid \phi_i)$$

e.g. $\phi_i = (\mu_i, \sigma_i^2)$ in $\mathcal{N}(\mu_i, \sigma_i^2)$

$$h_\theta, g_\psi : \text{ Neural Networks}$$

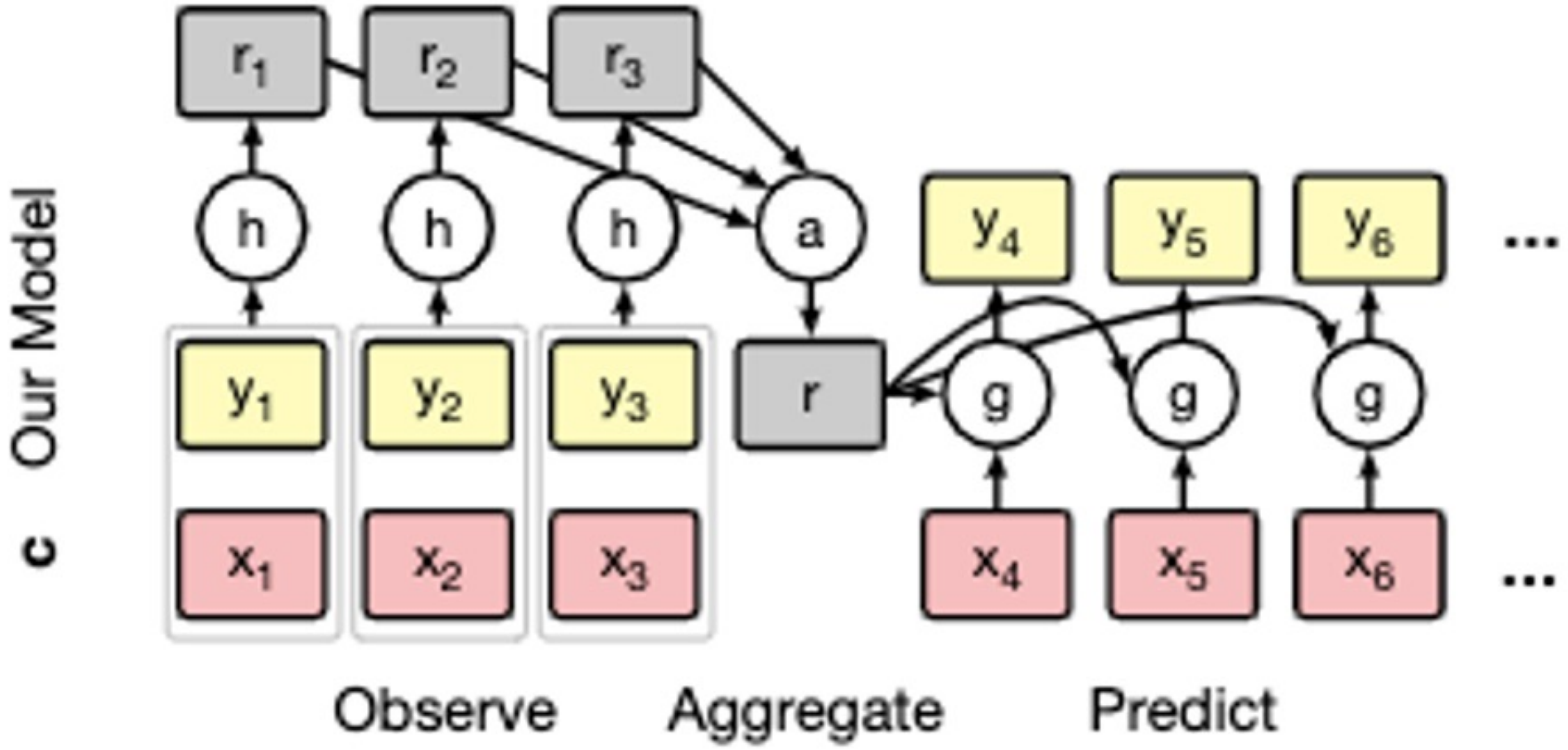
$$a(r_{1:N}) = \oplus : \text{ commutative operation}$$

$$\text{e.g. } r_1 \oplus r_2 \oplus \dots r_{n-1} \oplus r_n = \frac{1}{n}\sum_{i=1}^{n} r_i$$

- Points:
- **Why commutative ⊕?**
  - even if training data is permutated, we want same result
- **Flexibility to new (training) data**
  - for all observation, use the same encoder h and a
  - new training data can be encoded as well, thanks to commutativity and training data reuse

Conditional Neural Process

- architecture :
- formulation :



observations $O = \{(x_i, y_i)\}_{i=1}^{N} \subset X \times Y$

targets $T = \{x_i\}_{i=N+1}^{N+M} \subset X$

$$r_i = h_\theta(x_i, y_i) \quad \forall (x_i, y_i) \in O$$

representation vector : $r = r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n$

distribution parameter : $\phi_i = g_\psi(x_i, r) \quad \forall (x_i) \in T$

$$Q_{\theta\psi}(f(x_i) \mid O, x_i) = Q(f(x_i) \mid \phi_i)$$

e.g. $\phi_i = (\mu_i, \sigma_i^2)$ in $\mathcal{N}(\mu_i, \sigma_i^2)$

$$h_\theta, g_\psi : \text{ Neural Networks}$$

$$a(r_{1:N}) = \oplus : \text{ commutative operation}$$

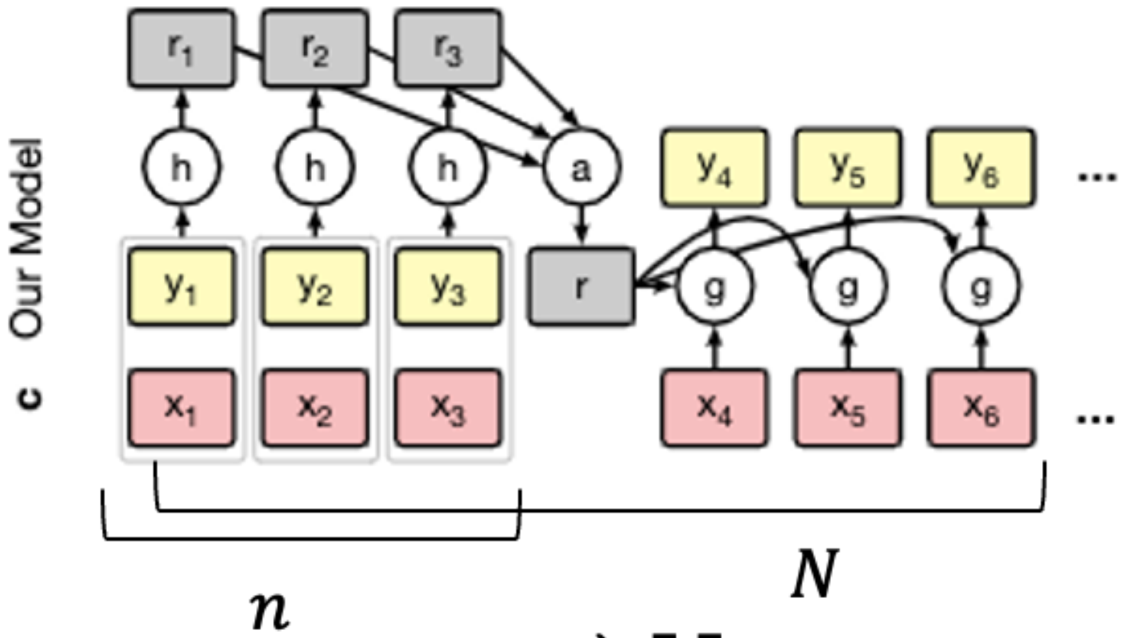
e.g. $r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n = \frac{1}{n}\sum_{i=1}^{n} r_i$

- Points:
- Probabilistic model
  - can consider the distribution of $f(x_i)$
- Can't provide (correct) joint distribution of $\boldsymbol{f}^* = (f(\boldsymbol{x}_{N+1}), f(\boldsymbol{x}_{N+2}), \ldots, f(\boldsymbol{x}_{N+M}))$
  - only provide distribution in each prediction point $f(\boldsymbol{x}_{N+i})$ (GP is not)
  - =using assumption of factorizability

$$Q_\theta(f(T) \mid O, T) = \prod_{x \in T} Q_\theta(f(x) \mid O, x)$$

Conditional Neural Process

in training



$(\theta, \psi) \rightarrow \theta$にまとめて表記

- **Training :**
  - likelihood loss : $\mathcal{L}(\theta) = -\mathbb{E}_{f \sim P}\left[\mathbb{E}_n\left[\log Q_\theta\left(\{y_i\}_{i=1}^{N} \mid O_n, \{x_i\}_{i=1}^{N}\right)\right]\right]$
  - In the training phase, varying the ratio of training data and test data
  - Approximate expectation by Monte Carlo estimate sampling f and n

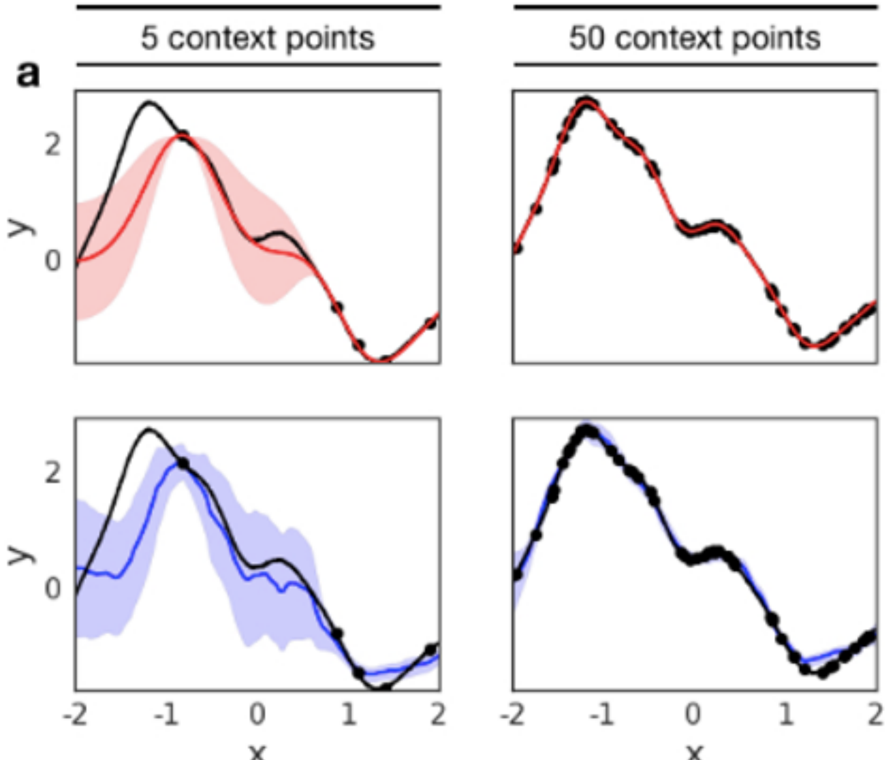
$$\boxed{\begin{array}{l} n \sim \text{uniform}\,[1, \ldots, N] \\ O_n = \{(x_i, y_i)\}_{i=1}^{n} \subset O \end{array}}$$

- **Experiment :**

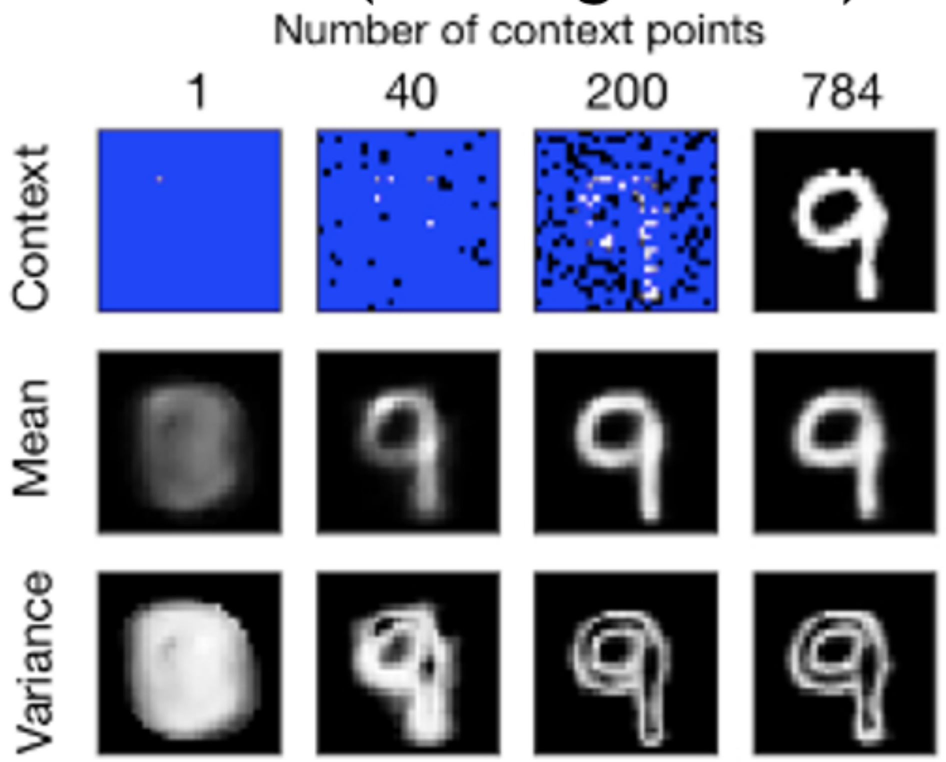
function sampled from GP (1-D regression)

GPs (expect best performance)

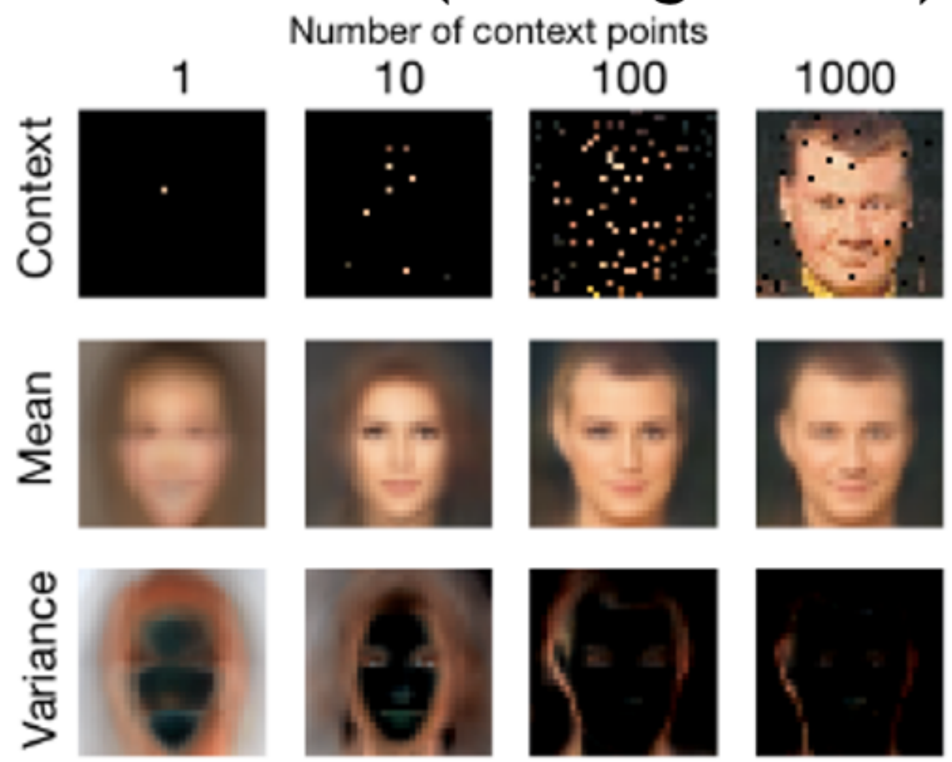
CNPs



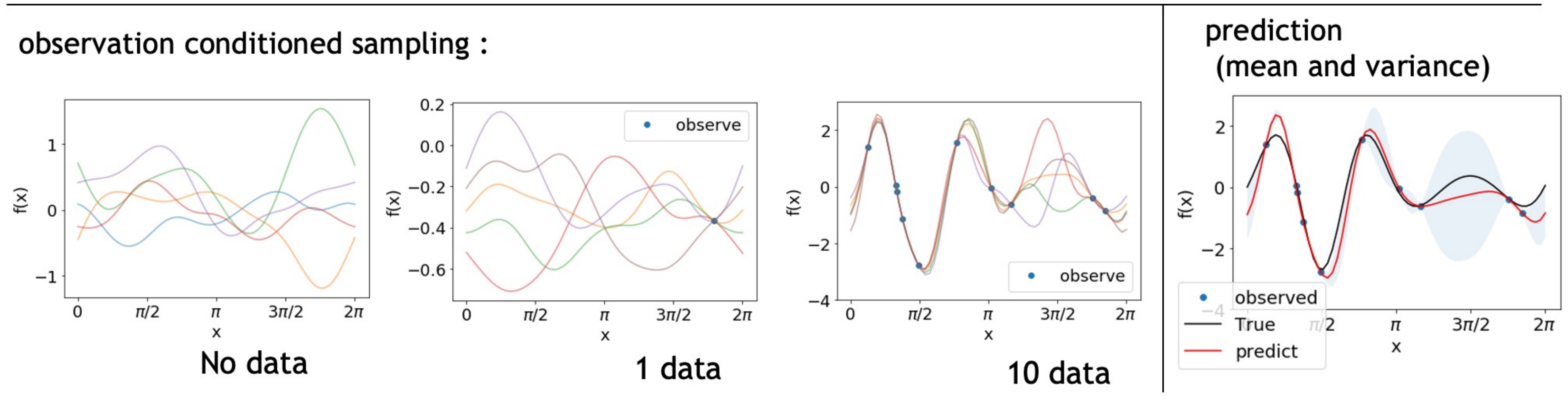
MNIST (2-D regression)



Celeb A (2-D regression)



再掲：GPからのサンプリング



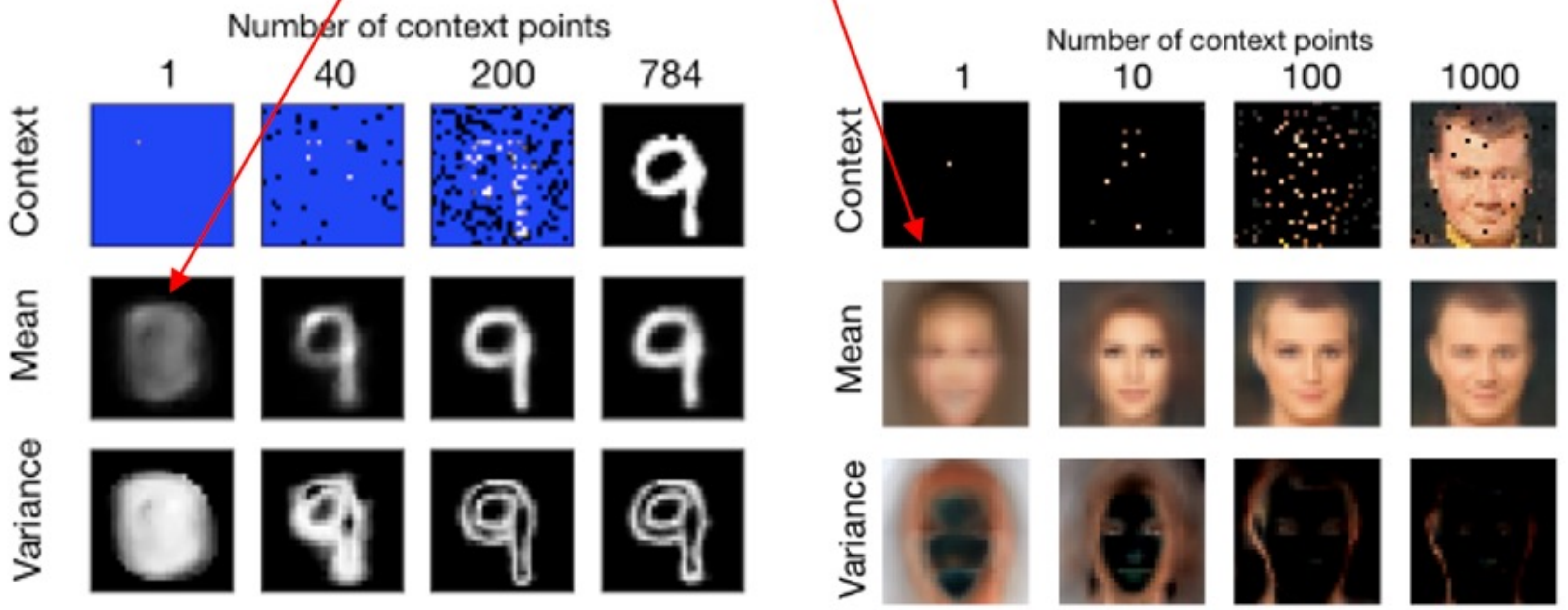
Coherence Problem

- CNPs can't get the correct joint distribution of $f^* = (f(x_{N+1}), f(x_{N+2}), \ldots, f(x_{N+M}))$, and only get distribution in each prediction point $f(x_{N+i})$, using assumption that there is no probablistic correlation between outputs

- Then the prediction, usually it seem best to use mean of distribution, don't have coherence (=not show the most likely one function, but show ambiguous layered functions)

- Can't sample a coherent output

  →Neural Process　で解決！

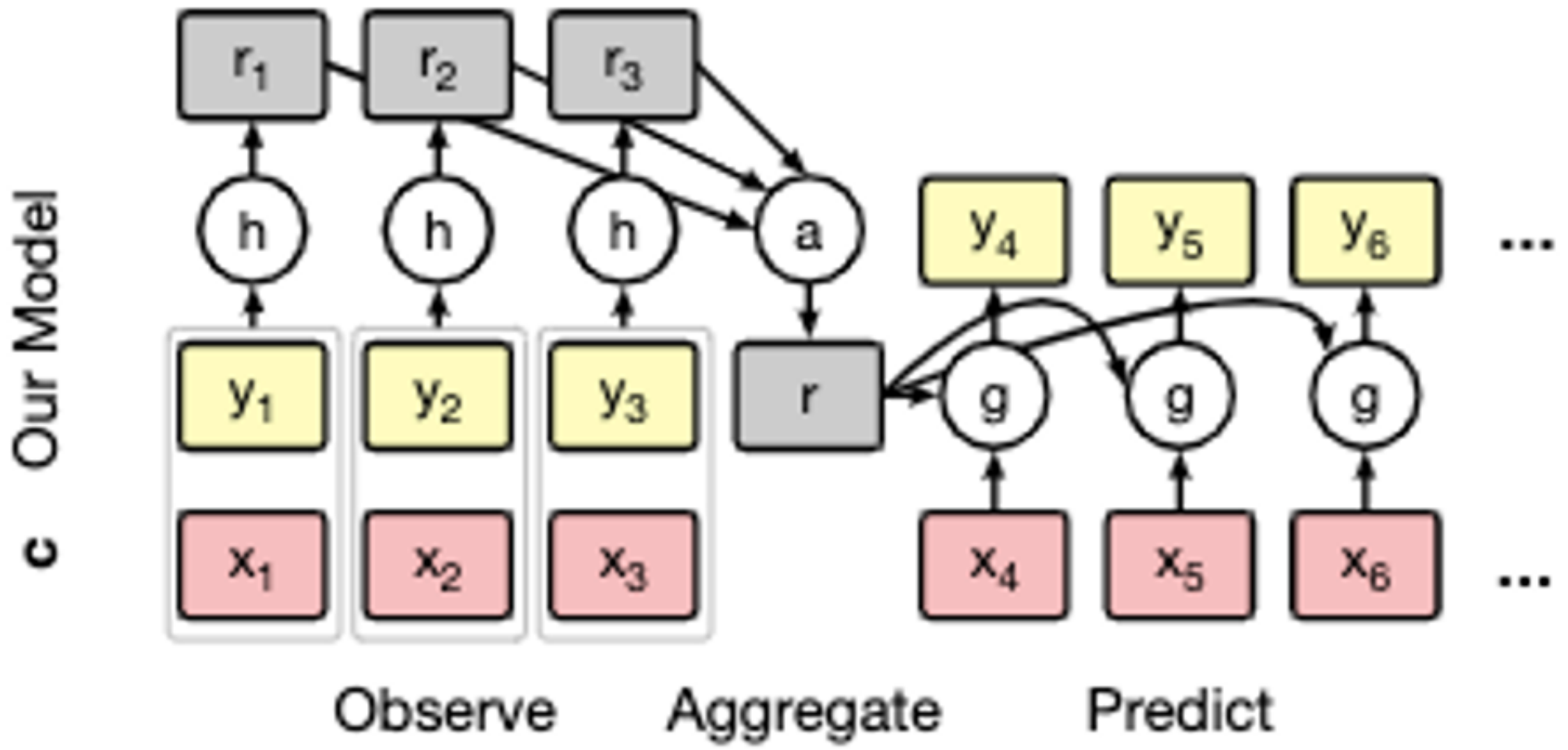


手法の説明2

showing again

Conditional Neural Process

- architecture :
- formulation :



observations $O = \{(x_i, y_i)\}_{i=1}^{N} \subset X \times Y$

targets $T = \{x_i\}_{i=N+1}^{N+M} \subset X$

$$r_i = h_\theta(x_i, y_i) \quad \forall (x_i, y_i) \in O$$

representation vector : $r = r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n$

distribution parameter : $\phi_i = g_\psi(x_i, r) \quad \forall (x_i) \in T$

$$Q_{\theta\psi}(f(x_i) \mid O, x_i) = Q(f(x_i) \mid \phi_i)$$

e.g. $\phi_i = (\mu_i, \sigma_i^2)$ in $\mathcal{N}(\mu_i, \sigma_i^2)$

$$h_\theta, g_\psi : \text{ Neural Networks}$$

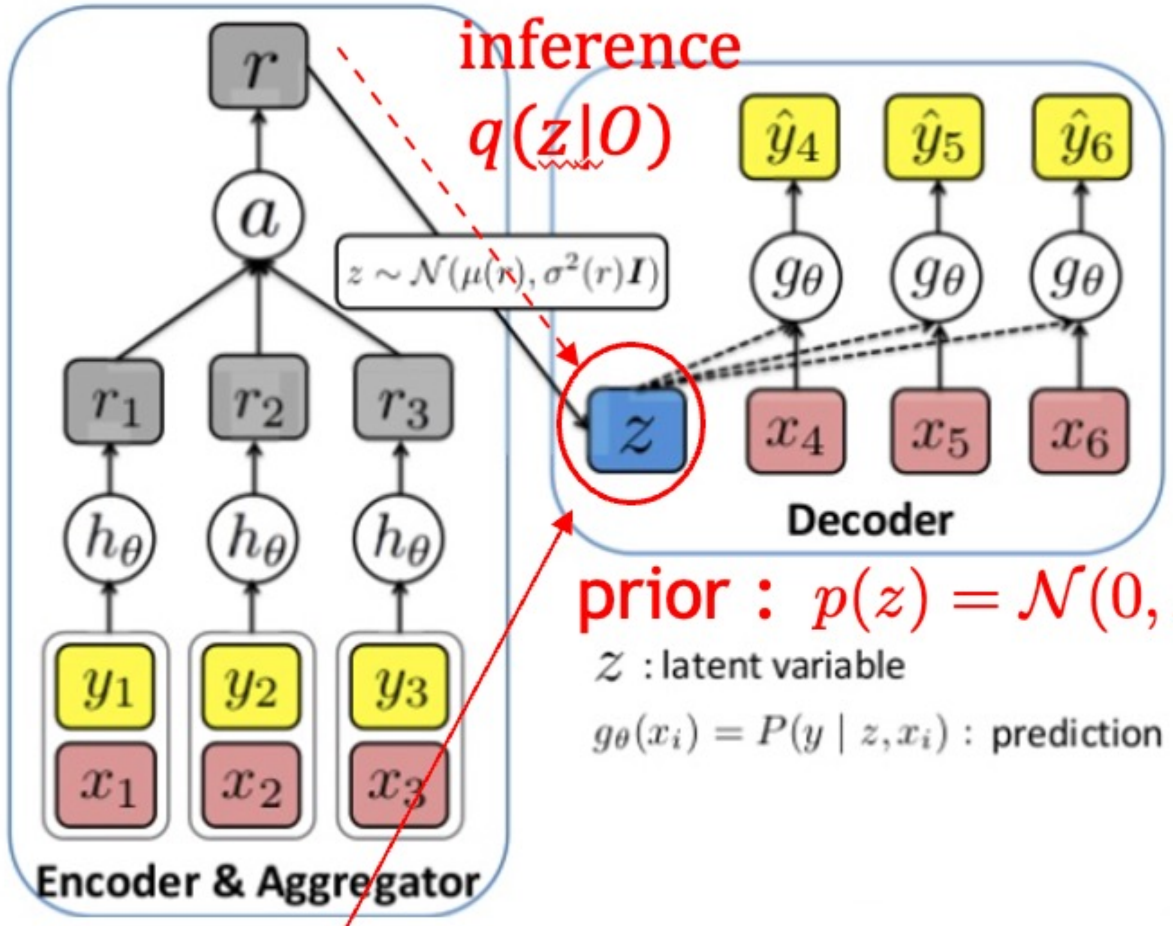
$$a(r_{1:N}) = \oplus : \text{ commutative operation}$$

$$\text{e.g. } r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n = \frac{1}{n}\sum_{i=1}^{n} r_i$$

- Points:

Neural Process

- architecture :



prior : $p(z) = \mathcal{N}(0, I)$

z : latent variable

$g_\theta(x_i) = P(y \mid z, x_i)$: prediction

- formulation :

observations $O = \{(x_i, y_i)\}_{i=1}^{N} \subset X \times Y$

targets $T = \{x_i\}_{i=N+1}^{N+M} \subset X$

$$r_i = h_\theta(x_i, y_i) \quad \forall (x_i, y_i) \in O$$

representation vector : $r = r_1 \oplus r_2 \oplus \dots r_{n-1} \oplus r_n$

latent variable: $z \sim \mathcal{N}(\mu(r), I\sigma(r))$

function value (random): $f(x_i) = g_\psi(x_i, z) \quad \forall (x_i) \in T$

h_θ, g_ψ : Neural Networks

$a(r_{1:N}) = \oplus$: commutative operation

e.g. $r_1 \oplus r_2 \oplus \dots r_{n-1} \oplus r_n = \frac{1}{n}\sum_{i=1}^{n} r_i$

- Points:
- Introduce latent variable $\boldsymbol{z}$
  - r is deterministic parameter, but $\boldsymbol{z}$ is random variable
  - $\boldsymbol{z}$ represent for randomness of function $\boldsymbol{f}$
  - so if given particuar $\boldsymbol{z}$, the function $\boldsymbol{f}$ have coherence
- Image of coherence

latent random variable : $\boldsymbol{z}$

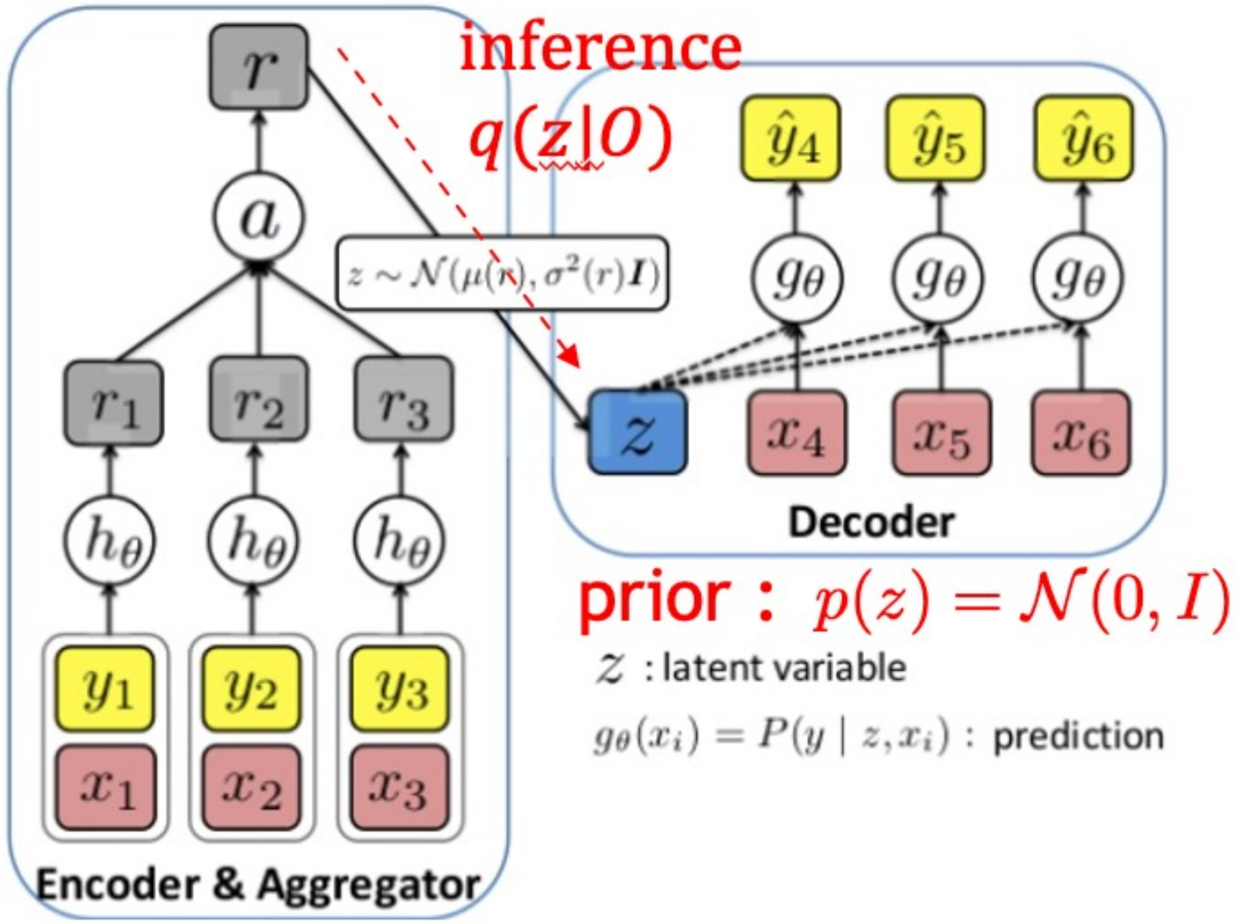
choose(generate)

$function_1$ $function_2$ $function_3$ . . .

. . .

Neural Process

- architecture :



prior : $p(z) = \mathcal{N}(0, I)$

z : latent variable

$g_\theta(x_i) = P(y \mid z, x_i)$: prediction

- formulation :

observations $O = \{(x_i, y_i)\}_{i=1}^{N} \subset X \times Y$

targets $T = \{x_i\}_{i=N+1}^{N+M} \subset X$

$$r_i = h_\theta(x_i, y_i) \quad \forall (x_i, y_i) \in O$$

representation vector : $r = r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n$

latent variable: $z \sim \mathcal{N}(\mu(r), I\sigma(r))$

function value (random): $f(x_i) = g_\psi(x_i, z) \quad \forall (x_i) \in T$

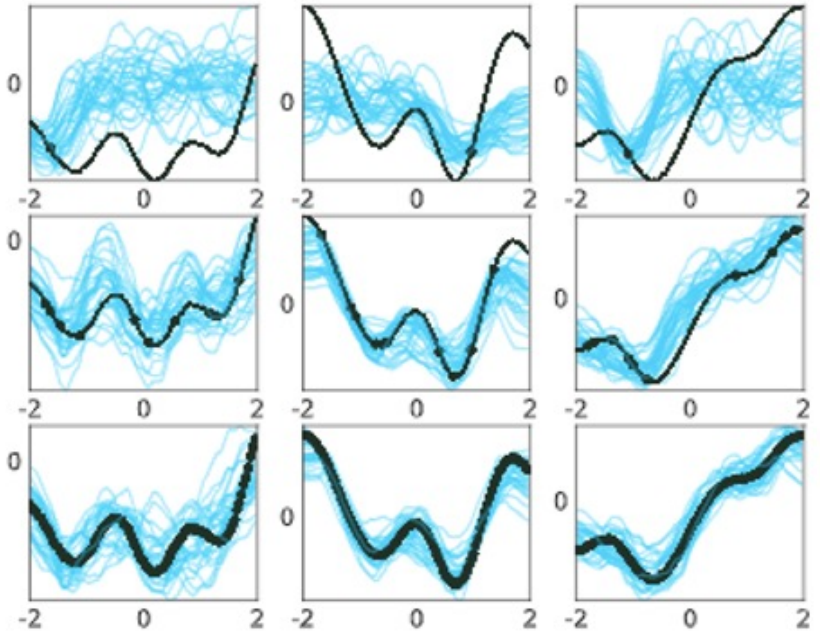
$$h_\theta, g_\psi : \text{ Neural Networks}$$

$$a(r_{1:N}) = \oplus : \text{ commutative operation}$$

$$\text{e.g. } r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n = \frac{1}{n}\sum_{i=1}^{n} r_i$$

- Points:
- Instead of coherence, can't analytically get prediction mean and variance (only sampling)
- we can get prediction by Monte Carlo method

1-D regression experiment



Neural Process

- architecture :
- formulation :

observations $O = \{(x_i, y_i)\}_{i=1}^{N} \subset X \times Y$

targets $T = \{x_i\}_{i=N+1}^{N+M} \subset X$

$$r_i = h_\theta\,(x_i, y_i) \quad \forall\,(x_i, y_i) \in O$$

representation vector : $r = r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n$

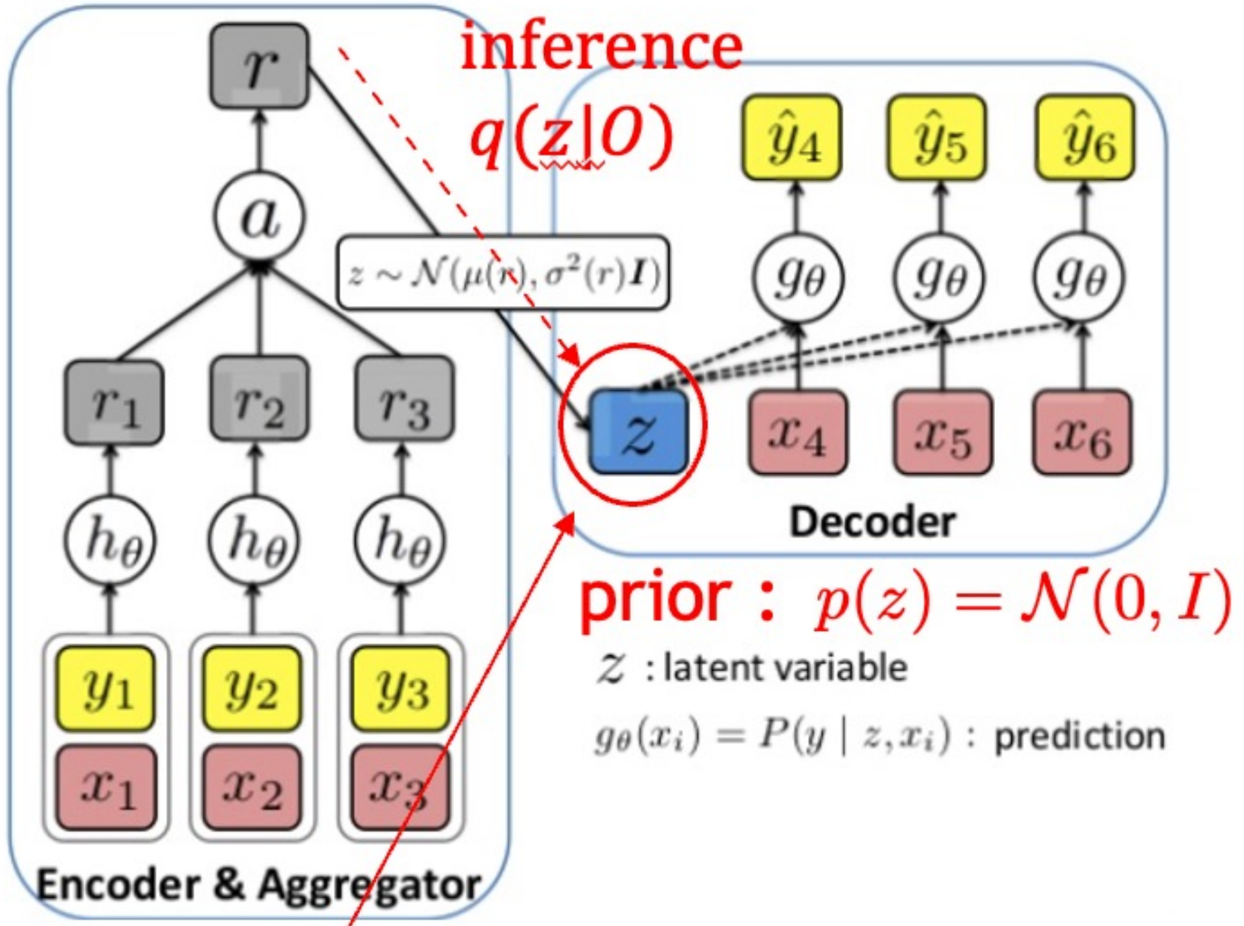
latent variable: $z \sim \mathcal{N}(\mu(r), I\sigma(r))$

function value (random): $f(x_i) = g_\psi\,(x_i, z) \quad \forall\,(x_i) \in T$

$$h_\theta, g_\psi \;:\; \text{Neural Networks}$$

$$a(r_{1:N}) = \oplus \;:\; \text{commutative operation}$$

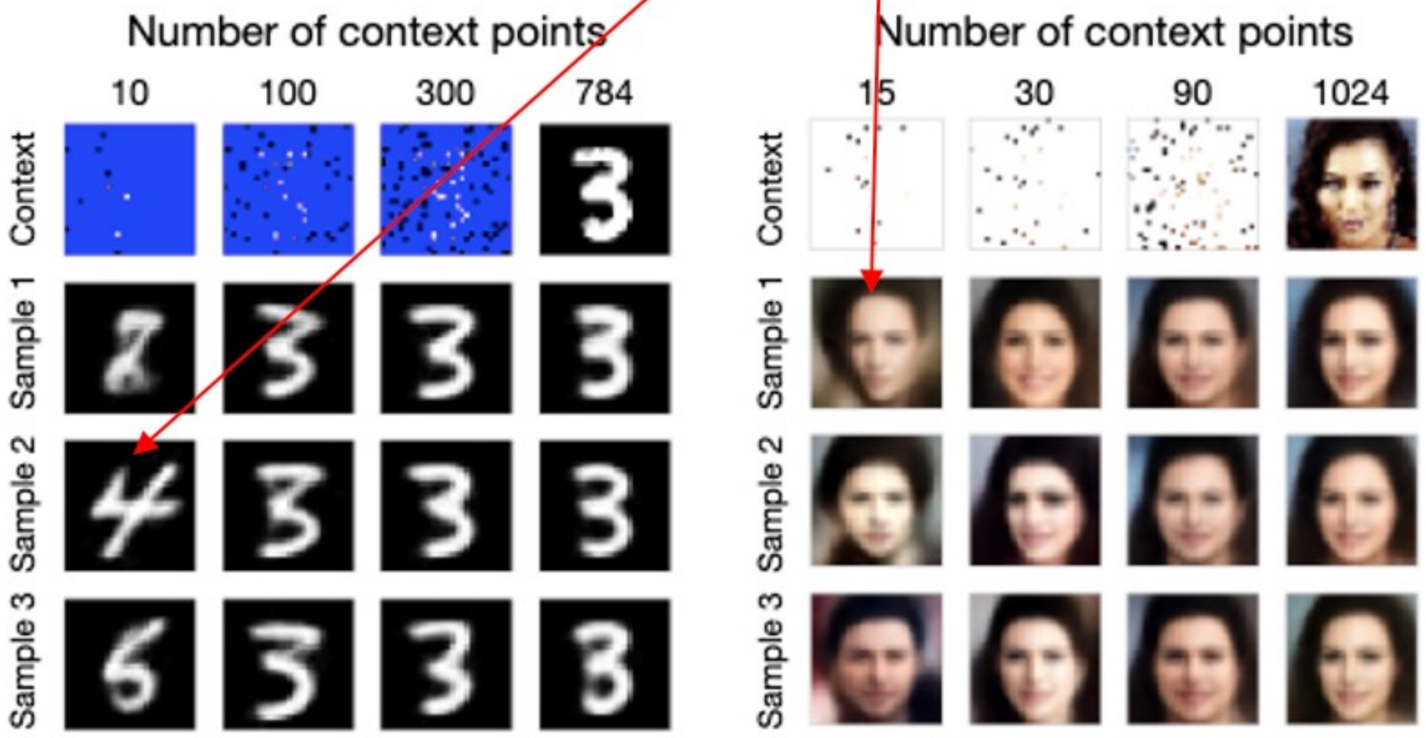
$$\text{e.g. } r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n = \frac{1}{n}\sum_{i=1}^{n} r_i$$



- Points:
- coherence

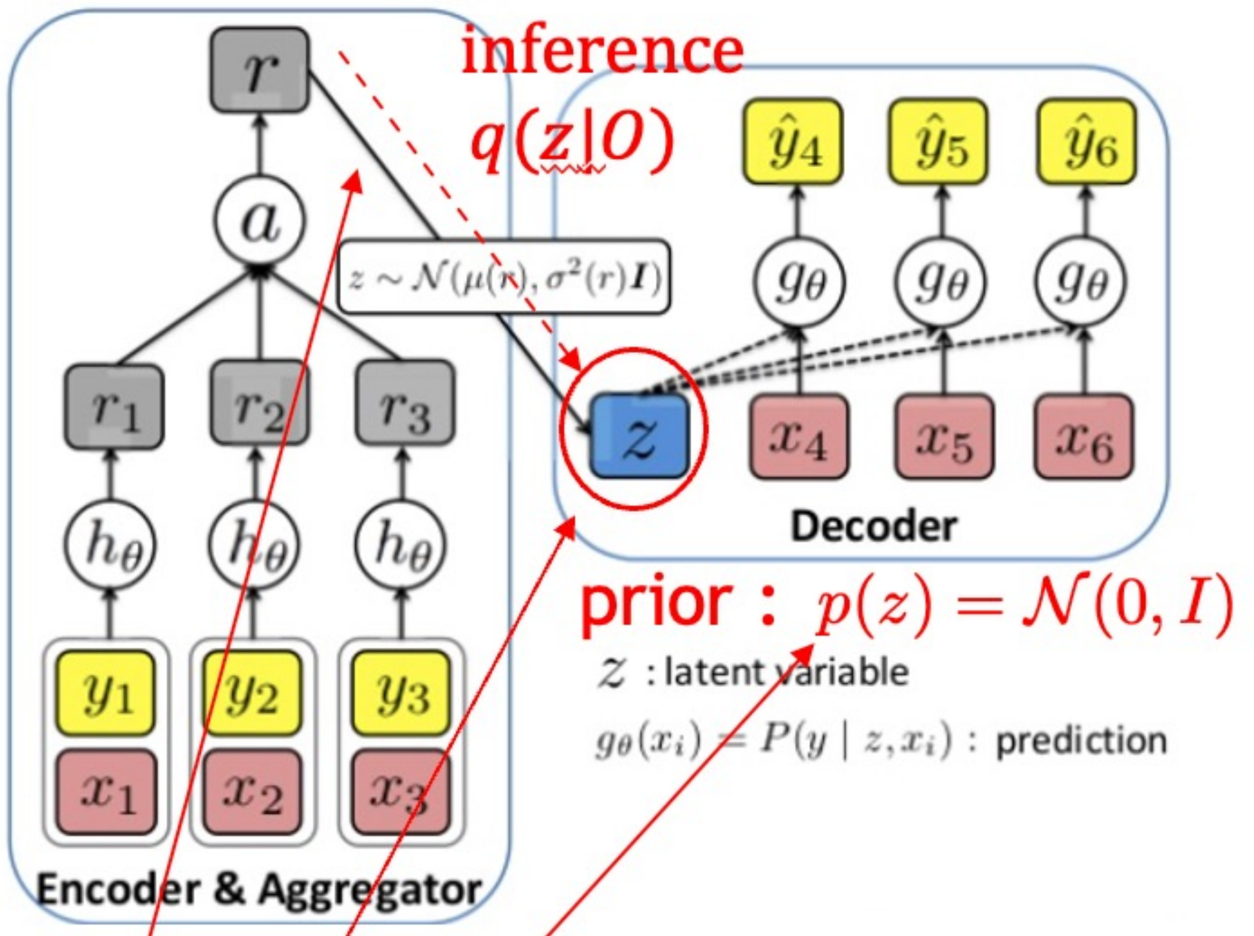
less ambiguous

same experiments



Neural Process

- architecture :



- formulation :

observations $O = \{(x_i, y_i)\}_{i=1}^{N} \subset X \times Y$

targets $T = \{x_i\}_{i=N+1}^{N+M} \subset X$

$$r_i = h_\theta(x_i, y_i) \quad \forall (x_i, y_i) \in O$$

representation vector : $r = r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n$

latent variable: $z \sim \mathcal{N}(\mu(r), I\sigma(r))$

function value (random): $f(x_i) = g_\psi(x_i, z) \quad \forall (x_i) \in T$

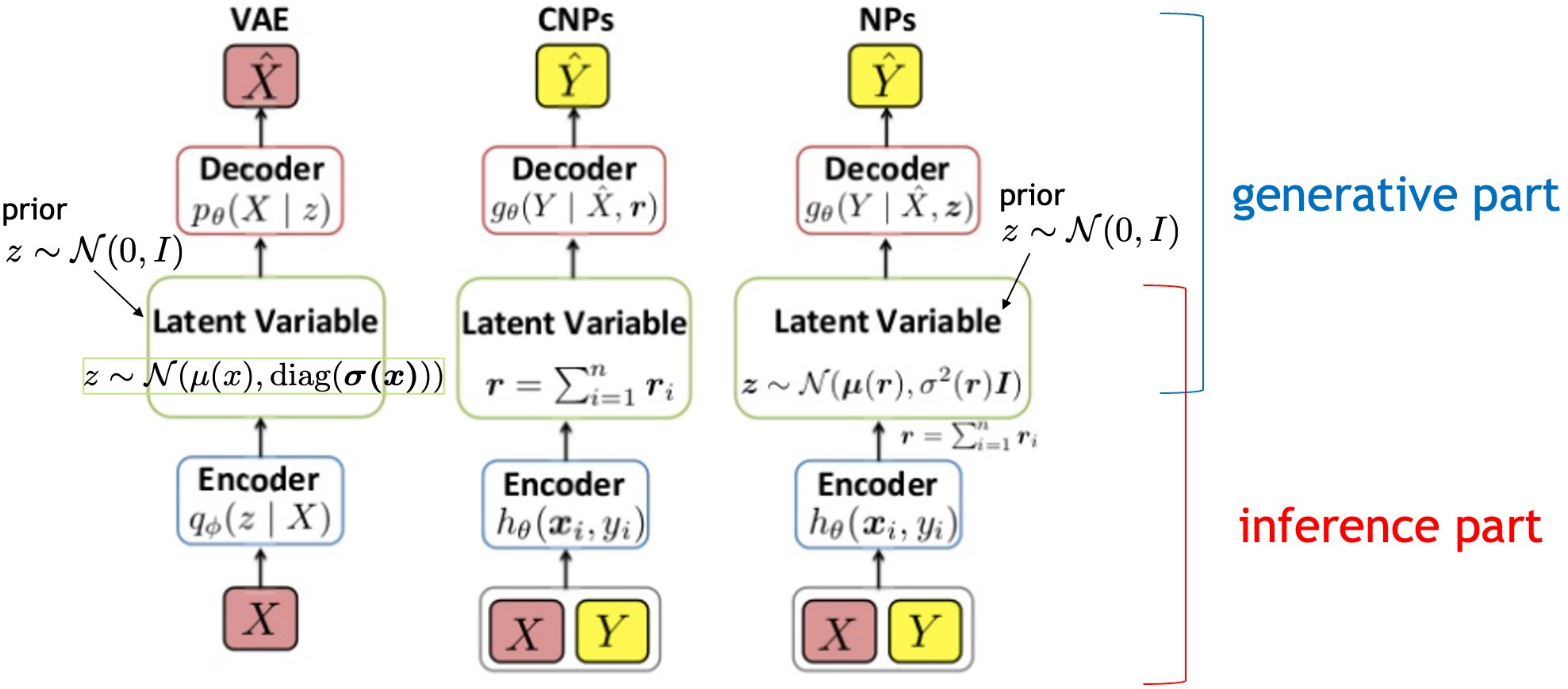
$$h_\theta, g_\psi : \text{ Neural Networks}$$

$$a(r_{1:N}) = \oplus : \text{ commutative operation}$$

$$\text{e.g. } r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n = \frac{1}{n}\sum_{i=1}^{n} r_i$$

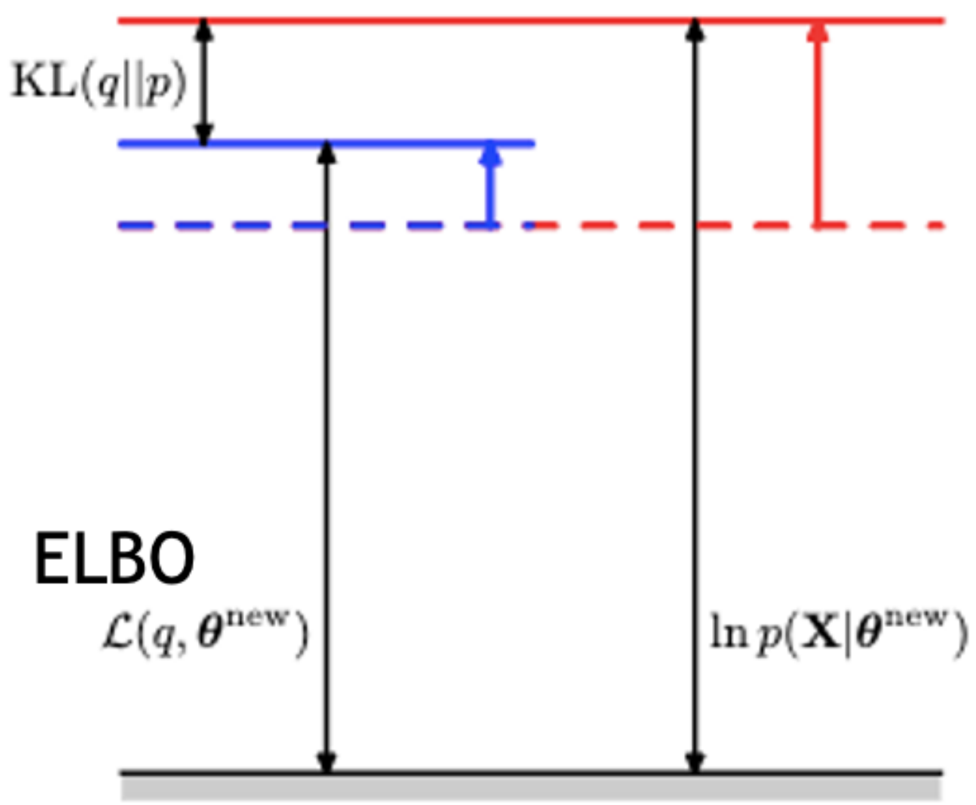
- Points:
- Using the idea of VAE
  - they are very similar architecture (next slide)
  - in both of model, loss form is based on theorem of <u>amortized variational inference</u>

Neural Process and VAE



Amortized Variational Inference

variational inference image



- Sorry for not touching detail

- In NPs (or VAE), model gives prior $p(z)$ and $p(y|z)$ (in VAE : $p(x|z)$)
- but posterior $p(z|y) = \frac{p(y|z)p(z)}{\int p(y|Z)p(z)dz}$ is intractable (because $p(y|z)$ is DNN modeled, integration is difficult)

- so use alternative distribution $q_\theta(z|y)$ (modeled by DNN), and farame work of variational inference

for any $q(z)$

instead of log likelihood, maximize ELBO (Evidence Lower BOund)

$$\log p_\theta(\mathbf{y}) = \underline{\mathcal{L}(q, \theta; \mathbf{y})} + D_{KL}\left[q(\mathbf{z}) \| p_\theta(\mathbf{z} \mid \mathbf{y})\right]$$

$$\left(\mathcal{L}(q, \theta; \mathbf{y}) = \int q(\mathbf{z}) \log \frac{p_\theta(\mathbf{y}, \mathbf{z})}{q(\mathbf{z})} d\mathbf{z} \right)$$

- in Amortized Variational Inference, $q_\theta(z|y)$ as $q(z)$

Objective Function of NPs

- Given observation, as training data, $O = \{(x_i, y_i)\}_{i=1}^{N} \subset X \times Y$,
  and then, for prediction training, we split N data into data of 1:n and data of $n+1$:N .

resulting objective function, ELBO is like bellow :

$q\left(z \mid x_{1:n}, y_{1:n}\right)$

$$\log p\left(y_{n+1:N} \mid x_{1:N}, y_{1:n}\right)$$

$$\geq \mathbb{E}_{q(z|x_{1:N}, y_{1:N})}\left[\underbrace{\sum_{i=n+1}^{N} \log p\left(y_i \mid z, x_i\right)}_{\text{reconstruction error}} + \underbrace{\log \frac{p\left(z \mid x_{1:n}, y_{1:n}\right)}{q\left(z \mid x_{1:N}, y_{1:N}\right)}}_{\text{generalization error}}\right]$$

- part is hard to analytically calculate, so approximate using $q(z|x_{1:N}, y_{1:N})$

- Expectation is approximated by Monte Carlo estimation (same as VAE) advance) using reparameterization trick for back propagation

$$\mathbf{z} = \boldsymbol{\mu}(\boldsymbol{r}) + \sigma(r)\boldsymbol{\varepsilon}, \text{ where } \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \mathbf{I})$$