Conditional Neural Processes

2020. 02. 13

SeulGi Hong

seulgihong@kaist.ac.kr

Contents

- Introduction
- Concept
- Method
- Experimental Results
- TODO

Introduction

Combine the benefits of **Stochastic Processes** and **Deep Neural Networks**

- Neural Network
  - structured format
  - trained by gradient descent

- Stochastic Process (e.g. GP)
  - defines the distribution of functions

What is CNP?

CNP is a conditional distribution over functions

which is trained to model the empirical conditional distributions of functions f ~ P

Concept of Conditional Neural Process Q_θ

Define **conditional distribution over functions** given a set of observations.

- **observations are parameterized** by the **NNs**
- with fixed dimension

$$Q_\theta(f(x_i)\boxed{|O, x_i)} = Q(f(x_i)\boxed{|\phi_i})$$

How?

⇒ Encoder-Decoder Model

$$r_i = h_\theta(x_i, y_i) \qquad \forall (x_i, y_i) \in O \qquad \textbf{Encoder}$$

$$r = r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n \qquad \textbf{Aggregator}$$

$$\phi_i = g_\theta(x_i, r) \qquad \forall (x_i) \in T \qquad \textbf{Decoder}$$

MLILAB Machine Learning & Intelligence

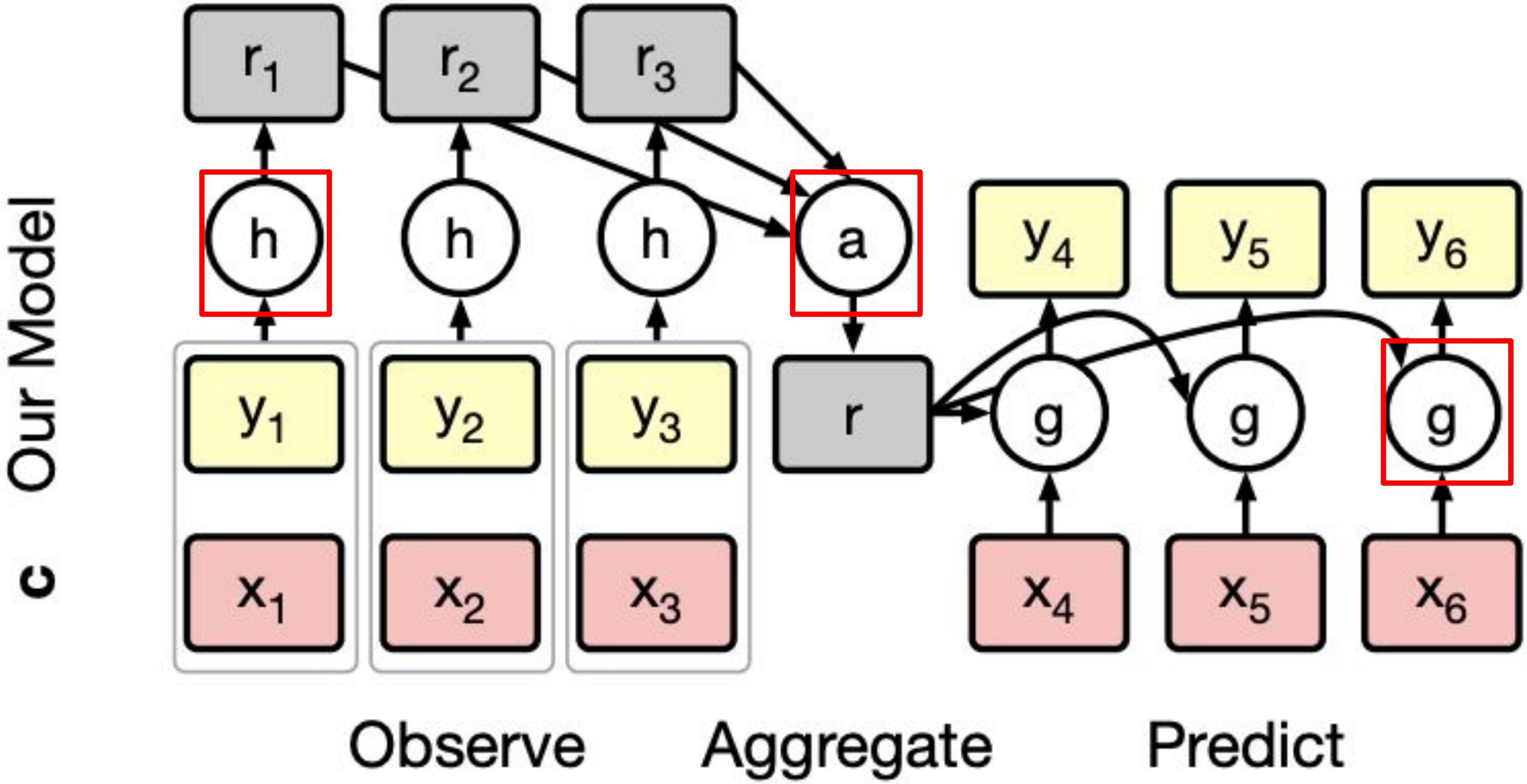
Concept of CNPs

- **Shifts the burden of imposing** **prior knowledge** from an analytic prior **to empirical data**

- Invariant under permutations of its inputs.
  Specifically, we use factored distribution in this paper

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

Method



c Our Model
r1
r2
r3
h
h
h
a
y4
y5
y6
y1
y2
y3
r
g
g
g
x1
x2
x3
x4
x5
x6
Observe
Aggregate
Predict

h function

- input [N, x_dim] and [N, y_dim] **; concat**
- output [N, r_dim]
- any operation to extract features (e.g. MLP, convolutions…)

O(N)

aggregator

- input [N, r_dim] ⟶ output [r_dim]
- commutative operation (e.g. average)
- **compression of observed knowledge**
  (just think about mean in GP)

O(1)

g function & model output

g function

- input [r_dim] and [M, x_dim]
- output [M, phi_dim] ; parameterized form

Regression task

- use phi to parameterize the **mean** and **variance** of Gaussian distribution

Classification task

- use phi to parameterize the **logits of the class probability p_c**

O(M)

CNP - How to Train

Loss : Negative conditional log probability

$$L(\theta) = -E_{f\sim P}\left[E_N\left[logQ_\theta\left(\boxed{\{y_i\}_{i=0}^{n-1}}\,|\,\boxed{O_N,}\,\boxed{\{x_i\}_{i=0}^{n-1}}\right)\right]\right]$$

Randomly chosen subset of O $\quad O_N = \{(x_i, y_i)\}_{i=0}^{N} \subset O$

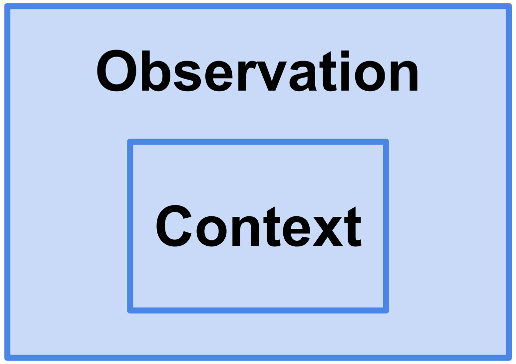
Set of Observations $\quad O = \{(x_i, y_i)\}_{i=0}^{n-1}$

CNP - Train stage

Encoder

- input : context x, y pair **(part of the training data)**
- output : context vector r

Train data



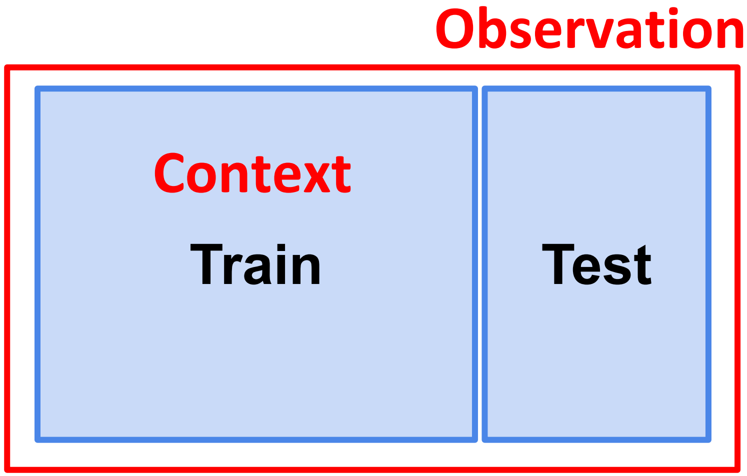
Decoder

- input : observation x **(all of the training data)** & context vector r
- output : mean & variance of all observation x

CNP - Test Stage

Encoder

- input : context x, y pair **(all of the training data)**
- output : context vector r



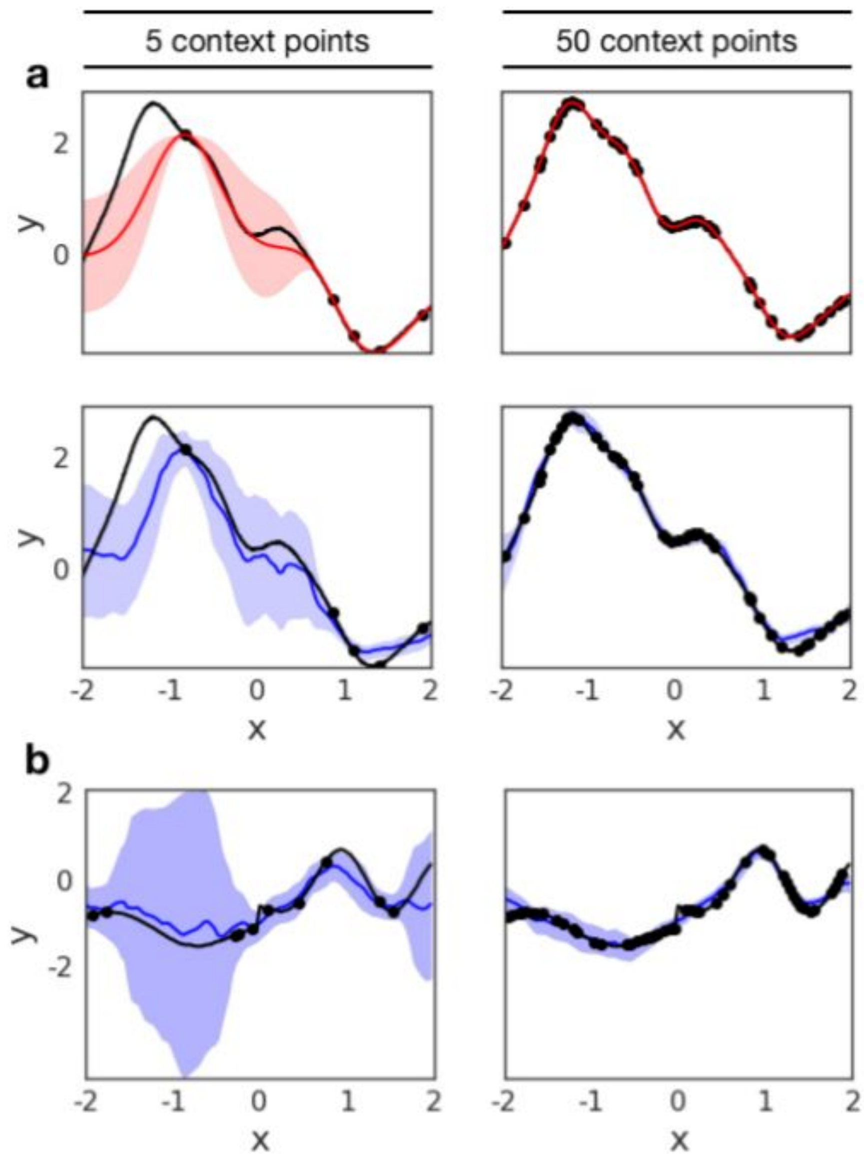
Decoder

- input : observation x **(train + test data)** & context vector r
- output : mean & variance of all observation x

Experimental Results

GP vs CNP

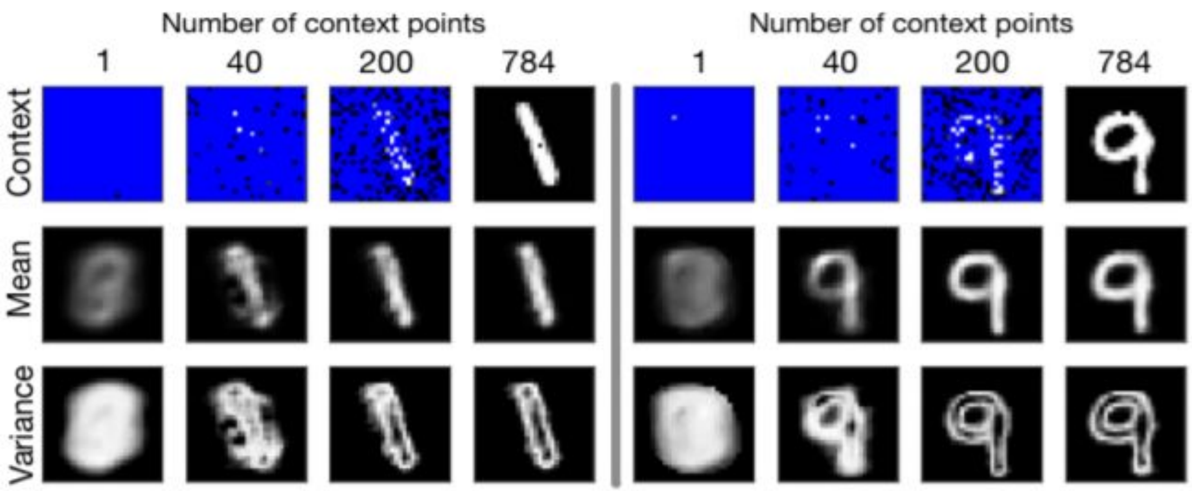
1D Regression



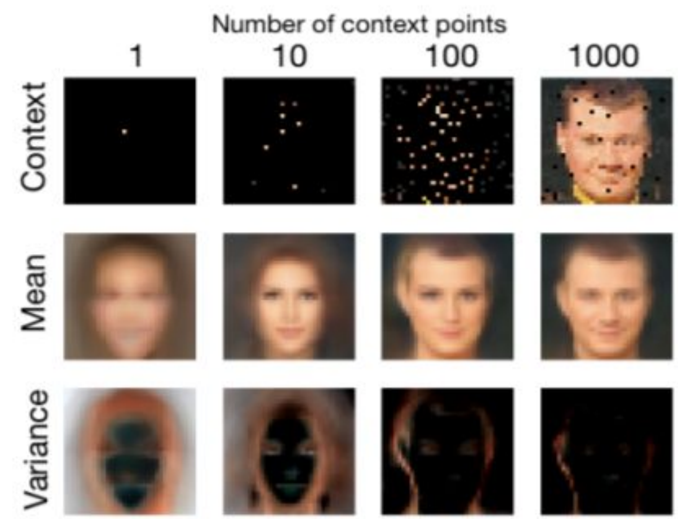
Experimental Results

2D Regression (image completion)

MNIST



CelebA



#	Random Context 10	Random Context 100	Random Context 1000	Ordered Context 10	Ordered Context 100	Ordered Context 1000
kNN	0.215	0.052	0.007	0.370	0.273	0.007
GP	0.247	0.137	**0.001**	0.257	0.220	**0.002**
CNP	**0.039**	**0.016**	0.009	**0.057**	**0.047**	0.021

Experimental Results

Classification on Omniglot

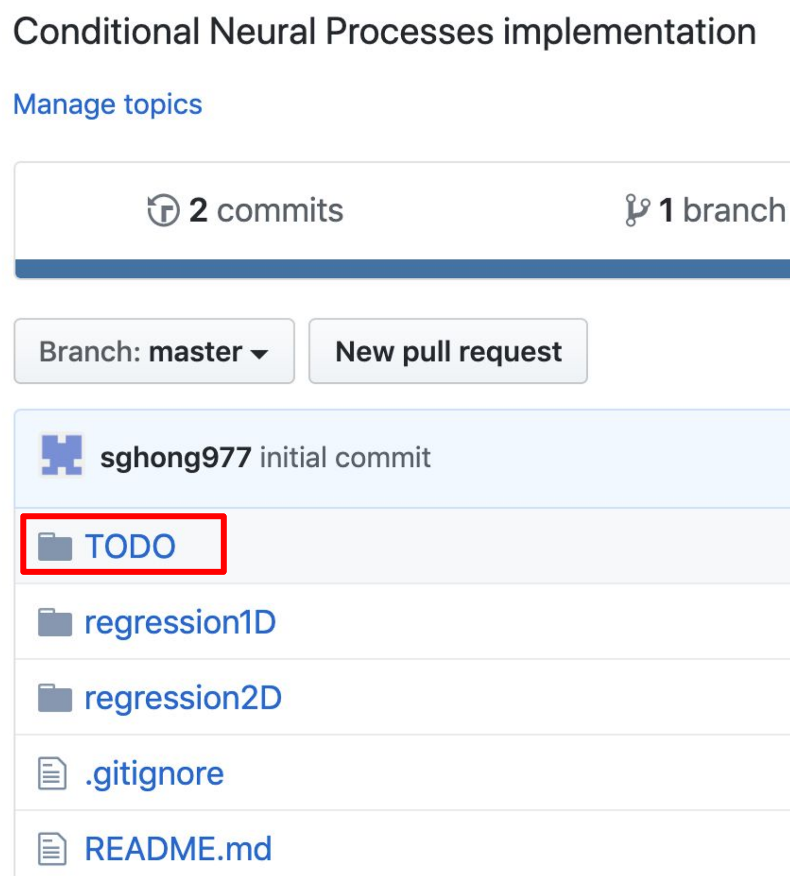
→ achieving comparable performance with inexpensive computational cost

	5-way Acc		20-way Acc		Runtime
	1-shot	5-shot	1-shot	5-shot	
MANN	82.8%	94.9%	-	-	$\mathcal{O}(nm)$
MN	**98.1%**	**98.9%**	**93.8%**	**98.5%**	$\mathcal{O}(nm)$
CNP	95.3%	98.5%	89.9%	96.8%	$\mathcal{O}(n+m)$

Table 2. **Classification results on Omniglot**. Results on the same task for MANN (Santoro et al., 2016), and matching networks (MN) (Vinyals et al., 2016) and CNP.

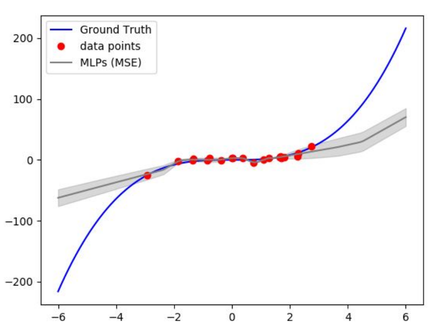
Implement CNP for 1D Regression task

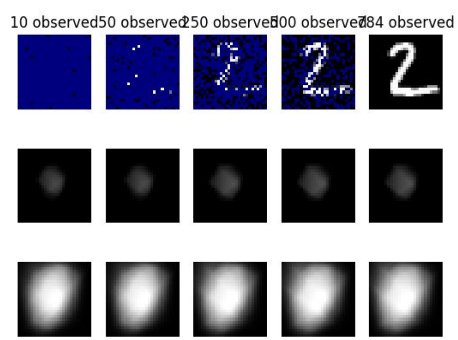
Link : https://github.com/sghong977/week6_CNP



- **1D regression task**

- (optional)
  2D regression task (image completion)





TODO

```python
# --------- TRAIN ---------
for epoch in range(epochs):
    nll_loss =  0.0
    optimizer.zero_grad()

    # TODO -- Train model

    # -------

    if epoch % print_step == 0:
        print('Epoch', epoch, ': nll loss', nll_loss.item())
    #dot = make_dot(mean)
    #dot.render("model.png")

    nll_loss.backward()
    optimizer.step()

print("final loss : nll loss", nll_loss.item())
result_mean, result_var = None, None

# TODO -- Calculate result_mean and result_var using your model

# -------

draw_graph(x_test, y_test, x_train, y_train, result_mean, np.sqrt(result_var))
```

Train

Validation

```python
# TODO -- Implement CNP model for 1d regression

class CNPs(nn.Module):
    def __init__(self, encoder, decoder):
        super().__init__()
        self.encoder = encoder
        self.decoder = decoder

    # Output : mean, variance
    def forward(self, x_ctx, y_ctx, x_obs):
        pass

class Encoder(nn.Module):
    def __init__(self, x_dim=1, y_dim=1, r_dim=128):
        super().__init__()
        pass

    def forward(self, x, y):
        pass

class Decoder(nn.Module):
    def __init__(self, x_dim, r_dim, y_dim):
        super().__init__()
        pass

    def forward(self, x, r):
        pass
```

Model