Survey on Neural Processes

Minjoo Choi[1]

Abstract

A Neural Process (NP) was appeared for the purpose of fast prediction and getting uncertainty recently. The traditional network for uncertainty is a Gaussian process but it severely suffers from inference-time efficiency. Similarly, the NP learns a stochastic process using observed data directly for functional uncertainty with much faster test time. To achieve a purpose, NP combines a neural network and a Gaussian process but there are no strong assumptions on the model aside from must-have properties to become a stochastic process. In this project, we will review on the Neural Process Family and its advanced application literatures for robustness and sequential data.

1. Introduction

There are two domain which conventional deep learning cannot deal with well. First one is 1) small data regime and the other is 2) where good uncertainty estimation is required. Meta-learning tries to solve a sparse data regime and Gaussian process does uncertainty estimation. However, Both of them have weakness in training or test time and prediction robustness.

Neural Process Family (**NPF**) tries to solve those issues by meta-learning those predictor's distribution and a stochastic process for getting a functional distribution. Then corresponding problems become:

- How can Neural Process parameterise the mapping relation between random targets and predictive distribution by neural networks?
- How to learn parameters of networks?

Hence, every topics related to NPF is how to formulate those two problems exactly. We surveyed five literatures about NPF and organised how to improve its robustness and sequential data processing.

[1]Department of Industrial Engineering, UNIST, Korea. Correspondence to: <mincule@unist.ac.kr>.

The overall structure of NPF is as follows.

NPF maps a context set C into representation R using an encoder Enc_θ. Specifically, the encoder is:

$$Enc_\theta(\mathcal{C}) = \rho(\Sigma_c^C \phi(x^{(c)}, y^{(c)}))$$

where ρ and ϕ are neural networks. The architecture of encoder could be selected by practitioners' inductive bias preference.

NPF is categorised into two sub-family depending on whether representation is used to define a stochastic latent variable.

1. Conditional Neural Process Family (CNPF)

A predictive distribution for targets is factorised conditioned on representation R directly.

$$p_\theta(\mathbf{y}_\mathcal{T}|\mathbf{x}_\mathcal{T}; \mathcal{C}) = \Pi_{t=1}^T p_\theta(y^{(t)}|x^{(t)}, R)$$

2. Latent Neural Process Family (LNPF)

Representation R is used to define a global latent variable $\mathbf{z} \sim p_\theta(\mathbf{z}|R)$. Then the predictive distribution is factorised conditioned on $\mathbf{z}$.

$$p_\theta(\mathbf{y}_\mathcal{T}|\mathbf{x}_\mathcal{T}; \mathcal{C}) = \int \Pi_{t=1}^T p_\theta(y^{(t)}|x^{(t)}, \mathbf{z}) p_\theta(\mathbf{z}|R) d\mathbf{z}$$

It depends on the situation which family we should use.

2. Preliminary

In this section, we will introduce two important concepts which are the backbone of NPs. We will see definition of a stochastic process and consider sufficient properties in order to be the stochastic process. Since NP is to learn the stochastic process from data directly, we must clarify those definition and properties. Preliminary requires a knowledge about a probability space and a measure. [1]

[1]Everything in this section was referred to https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.110/lehre/ws13/Stochastik_II/Skript_1.pdf

2.1. Stochastic process

Definition 2.1. Let $(\Omega, \mathcal{F}, P)$ be a probability space and let T be an arbitrary set (called the index set). Any collection of random variables $X = \{X_t : t \in T\}$ defined on $(\Omega, \mathcal{F}, P)$ is called a stochastic process with index set T.

A ramdom variable has its distribution (or can be described by a distribution). Since a stochastic process is a collection of random variables, it can be represented by a certain distribution. The below shows the distribution (or a measure) of the stochastic process.

Let $\{X_t : t \in T\}$ be a stochastic process with index T. Take some $t_1, \cdots, t_n \in T$. For Borel sets $B_1, \cdots, B_n \subset R$, define a probabillity measure such that $P_{t_1,\ldots,t_n}(B_1 \times \cdots \times B_n) = P[X_{t_1} \in B_1, \cdots, X_{t_n} \in B_t]$ or $P_{t_1,\cdots,t_n}(B) = P[(X_{t_1}, \cdots, X_{t_n}) \in B], B \subset R^n(Borel)$.

This probability measure on R^n is the distribution of the random variables $(X_{t_1}, \cdots, X_{t_n})$. It is often called a finite-dimensional distribution of X.

2.2. Kolmogorov's extension theorem

Theorem 2.2. *Fix any non-empty set T. Let $\mathcal{P} = \{P_{t_1,\cdots,t_n} : n \in N, t_1, \cdots, t_n \in T\}$ be a collection of probability measures which has the properties of **permuation invariance** and **projection invariance**. Then there exist a probability space $(\Omega, \mathcal{F}, P)$ and a stochastic process $\{X_t : t \in T\}$ on $(\Omega, \mathcal{F}, P)$ whose finite-dimensional distributions are given by the collection $\mathcal{P}$.*

Theorem 2.2 means that for every $n \in N$ and every $t_1, \cdots, t_n \in N$ the distribution of the random vector $(X_{t_1}, \cdots, X_{t_n})$ coincides with $P_{t_1,\cdots,t_n}$ (a finite-dimensinal distribution of X).

Hence, we understand the stochastic process and its sufficient construction properties. Those makes a NP modelled by data directly.

3. Models

We will examine five academic literatures in this part. First one is a Conditional Neural Process which is the beginning of Neural Process Family. And (Latent) Neural Process, more generalised version, is addressed. Others are related to the more specific problems such as underfitting, robustness and sequence. We've tried to retain notation in common for readers' understaning.

3.1. Conditional neural process

(Garnelo et al., 2018a) states two main problems in conventional supervised learning.

The conventional supervised learning requires:

1. A prior knowledge about an original mapping function f to get an approximate function g.
2. A ton of time and data to fit the curve (or the function g).

They devised a **Conditional neural process** (**CNP**) which produces predictive distributions over functions conditioned on a set of observations O. It is designed *to make a prior directly from the data* and *to be trained by gradient descent* where aims to maximise the conditional likelihood of an arbitrary subset of targets given an arbitrary subset of observations. This method leads to $\mathcal{O}(n + m)$ test time where n is the number of observations and m is the number of targets. (Total data $N = n + m$.)

Say an input $x_i \in X$ and an corresponding output $y_i \in Y$. Define:

$$O = \{(x_i, y_i)\}_{i=0}^{n-1} \subset X \times Y : Observations$$

$$T = \{x_i\}_{i=n}^{n+m-1} : Targets$$

Assume that the outputs y_i is a realisation of a **Stochastic Process** P, where $f \sim P$ such that $y_i = f(x_i)$.

Our task is to predict the outputs $f(x)$ for every $x \in T$ given O, which means to get a predictive distribution:

$$P(f(T)|O, T)$$

CNPs parameterise the distribution $f(T)$ conditioned on O via a fixed-vector representation r. It mimics the stochastic process but does not have consistency related to a prior process since it does not follow exact properties of a stochastic process. However, it has advantage in functional flexibility and scalability.

Now we formulate CNPs. A CNP is a conditional stochastic process Q_θ which is:

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

Here Q is invariant to permutations of O and T. θ is a parameter. ϕ_i is a combined-parameter rerpesentation of O and x_i. Refer to the later paragraph for more specific information.

This invariance makes Q factorise.

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

Next, we will make a representation of O. You can imagine an encoder-decoder architecture in the traditional deep learning domain.

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

$$r = r_1 \oplus r_2 \oplus \cdots \oplus r_n$$

$$\phi_i = g_\theta(x_i, r), \forall(x_i) \in T$$

where $h_\theta : X \times Y \to R^d$ and $g_\theta : X \times R^d \to R^e$ are neural networks, $\oplus$ is a commutative operation, and ϕ_i is a parameter and a representation condition. For each task, we can design ϕ_i properly and solve a regression or a classification task.

The objective function for training a CNP (or training Q_θ) is maximising:

$$\mathcal{L}(\theta) = -E_{f\sim P}[E_{N'}[\log Q_\theta(\{y_i\}_{i=0}^{n-1}|O_{N'}, \{x_i\}_{i=0}^{n-1})]]$$

where $O_{N'} = \{(x_i, y_i)\}_{i=0}^{N'} \subset O$ and $N' \sim uniform[0, \ldots, n-1]$.

To solve the function, we should use Monte Carlo estimates by sampling f and N'. We can conclude that it relies on the empirical data rather then a practitioner's prior knowledge.

However, CNPs have limitations in that factorisation ignores the covariance between target points. The Gaussian Process used a coregionalisation matrix to describe interaction between them. In Neural Process Family, it decided to use a latent variable instead.

3.2. Neural process

(Garnelo et al., 2018b) generalised a neural process conditioned on the fixed representation to the latent variable. Here, we need the Theorem 2.2 in preliminary to gaurantee the stochasticity of a **Neural Process** (**NP**).

Suppose that there is a finite-dimensional probability measure $P_{x_{1:n}}$ of the random vector $y_{1:n} = (y_1, \cdots, y_n)$ where $y_i = f(x_i)$ and $f : X \to Y$. And if this measure $P_{x_{1:n}}$ satisfies two properties mentioned in Kolmogorov's extension theorem, it is gauranteed that there exists a stochastic process $\{Y_t = F(x_t) : t \in T\}$. So, we will suppose two properties are hold. Say f is a realisation of one path of y_i corresponding to x_i. We can marginalise the probability by f:

$$P_{x_{1:n}}(y_{1:n}) = \int df P(y_{1:n}|f, x_{1:n})P(f)$$

Assume a Gaussian noise in observation y_i.

$$P_{x_{1:n}}(y_{1:n}) = \int df \Pi_{i=1}^n \mathcal{N}(y_i|f(x_i), \sigma^2)P(f)$$

Now, we parameterise the stochastic process function f by a random-vector z and learnable function g to approximate it by a neural network. Thus we have:

$$P(y_{1:n}, z|x_{1:n}) = \int dz \Pi_{i=1}^n \mathcal{N}(y_i|g(x_i, z), \sigma^2)P(z)$$

Training a NP uses the **evidence lower-bound**(**ELBO**). Split the N data into n observations and m targets ($N = n + m$). And approximate an intractable conditional prior $p(z|x_{1:n}, y_{1:n})$ to a tractable variational posterior $q(z|x_{1:n}, y_{1:n})$:

$$\log P(y_{n+1:N}|x_{1:n}, y_{1:n}, x_{n+1:N})$$
$$\geq E_{q(z|x_{1:N}, y_{1:N})}[\Sigma_{i=n+1}^N \log(P(y_i|x_i, z)) + \log \frac{q(z|x_{1:n}, y_{1:n})}{q(z|x_{1:N}, y_{1:N})}]$$

Simply,

$$\log P(y_T|O, x_T)$$
$$\geq E_{q(z|O,T)}[\Sigma_{(x_i,y_i)\in T} \log(P(y_i|x_i, z)) + \log \frac{q(z|O)}{q(z|O, T)}]$$

where $T = \{(x_i, y_i)\}_{i=n}^{n+m-1}$ means targets.

The NP consists of a encoder-decoder architecture with an aggregator. The aggregator is the same as that of CNP to satisfy the permutation invariance of a random variable of representation.

The main weakness of NP is that it suffers from underfitting.

3.3. Bootstrapping Neural process

A NP models point-wise uncertainty by a conditional distribution of y on x and this is related to the stochasticity in function realisations. Only a single Gaussian latent variable is enough to model the overall structure of functions is in question. The single latent variable can be a bottleneck of representation.

(Lee et al., 2020) introduces an alternative method for modeling functional uncertainty to the NP. It is a **bootstrap**. The bootstrap improves a model robustness in a model-data mismatch situation. This robustness exerts its effect in the *model-data mismatch* situation.

Let's see a naive **Residual Bootstrap**. Let $(X, Y) = \{(x_i, y_i)\}_{i=1}^n$ be a dataset. We will fix X and only sample the additive residuals ε between prediction and true value in order to prevent the data from under-sampling.

Say $y_i = \mu_i + \sigma_i\varepsilon_i$ where $P(y_i|x_i) = \mathcal{N}(y_i|\mu_i, \sigma_i^2)$. We will make a residual bootstrap dataset with replacement. Compute the residual $\varepsilon_i = \frac{y_i - \mu_i}{\sigma_i}$ from $\{(\mu_i, \sigma_i)\}_{i=1}^n$ of the dataset (X, Y). Then we have a residual set $\mathcal{E} = \{\varepsilon_i\}_{i=1}^n$. Instead of resampling in $\{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^n \sim_{s.w.r.} \{(x_i, y_i)\}_{i=1}^n$, we sample residuals $\tilde{\varepsilon}_1^{(j)}, \cdots, \tilde{\varepsilon}_n^{(j)} \sim_{s.w.r.} \mathcal{E}$ for $j \in [1, k]$ and construct a bootstrap dataset $\tilde{x}_i^{(j)} = x_i, \tilde{y}_i^{(j)} = \mu_i + \sigma_i\tilde{\varepsilon}_i^{(j)}$.

Next, authors designed a **Bootstrapping Neural Process** (**BNP**) to introduce the bootstrap context dataset with original context observations dataset in training step. Two contexts representations $\tilde{\phi}^{(j)} = f_{enc}(\hat{X}^{(j)}, Y^{(j)}), \phi = f_{enc}(X_c, Y_c)$ is feed to an adaptation layer $g(\cdot)$. Hence,

the BNP get a combined representation and introduce functional uncertainty.

An Ensembled predictive distribution of a BNP is:

$$P(y_i|x_i, X_c, Y_c) = \frac{1}{k}\Sigma_{j=1}^{k}\mathcal{N}(y_i|\mu_i^{(j)}, (\sigma_i^{(j)})^2)$$

Objective function of BNP is as follows:

$$E_{P(\mathcal{T})}[P(Y|X, C) + \log\frac{1}{k}\Sigma_{j=1}^{k}\mathcal{N}(y_i|\mu_i^{(j)}, (\sigma_i^{(j)})^2)]$$

The former term is a conventional ELBO and the latter is log likelihood of target Y.

The BNP gives the NP the *robustness*.

3.4. Sequential Neural Processes

A typical Neural Process (NP) cannot capture the underlying temporal dynamics of stochastic processes. Many real problems, especially in reinforcement learning, have temporal changes in their state functions, so introducing a temporal property in NP is a crucial task for modelling these problems.

(Singh et al., 2019) propose a **Sequential Neural Process** (**SNP**), which uses a state-space model to combine a temporal property in NP. The state-space model aims to get a state-transition probability $P(z_t|z_{t-1})$ which shows how previous states affect a current state. Here, SNP exploits a recurrent state-space model (Hafner et al., 2018) to capture the non-linear non-Markovian temporal dependency. Say h_t is a RNN hidden state and z_t is a stochastic latent state. They are updated as follows:

$$h_t = RNN(h_{t-1}, z_{t-1}), z_t \sim P(z_t|h_t), y_t \sim P(y_t|h_t, z_t)$$

Since h_t is related to the past latents $z_{<t}$, the recurrent state-space model is non-Markovian.

Imagine a situation where there are T stochastic processes $P_1, \cdots, P_T$. For each time step $t \in [1, T]$, the SNP aims to model a P_t conditioned on a latent z_t and a transition $P_{t-1} \rightarrow P_t$. Also, we have a context observation $C_t = \{(x_i^t, y_i^t)\}_{i\in\mathcal{I}(C_t)}$ which has a different quantity for each time step. (It could be zero.)

Now, we will see how to design the SNP. First, the relation of $P_{t-1} \rightarrow P_t$ corresponds to $P(z_t|z_{<t}, C_t)$ distribution. This acquired z_t is used to model the target label distribution $P(y_t|x_t, z_t)$. Say $C = (C_1, \cdots, C_T)$ for simple notation.

The predictive distribution of a SNP is:

$$P(Y, Z|X, C) = \Pi_{t=1}^{T}P(Y_t|X_t, z_t)P(z_t|z_{<t}, C_t)$$

where $P(Y_t|X_t, z_t) = \Pi_{i\in\mathcal{I}(D_t)}P(y_i^t|x_i^t, z_t)$ and $z_o = null$.

Same with the NP, training is conducted by variational approximation, i.e. ELBO. A true posterior is approximated to the factorised neural network, $P(Z|C, D) \approx \Pi_{t=1}^{T}Q_\phi(z_t|z_{<t}, C, D)$.

The following is the ELBO of SNP:

$$\begin{aligned}\log P(Y|X, C) &\geq \mathcal{L}_{SNP}(\theta, \phi)\\ &= \Sigma_{t=1}^{T}E_{Q_\phi(z_t|C,D)}[\log P_\theta(Y_t|X_t, z_t)]\\ -E_{Q_\phi(z_{<t}|C,D)}&[KL(Q_\phi(z_t|z_{z<t}, C, D)||P_\theta(z_t|z_{<t}, C_t))]\end{aligned}$$

Additionally, to mitigate a transition collapse problem which the latent $z_{<t}$ information becomes more prevalent than the context information C_t, authors introduced the posterior-dropout ELBO in objective function. But this is not the main content of the paper, we do not discuss it and refer to the paper for more information.

SNPs are regarded as a meta-transfer learning where temporal dynamics are represented as the latent states of RNN and these states affect the next stochastic process. So, latent states of each t step helps to transfer the previous knowledge.

3.5. Robustifying Sequential Neural Processes

A SNP can model the temporal dynamics of stochastic processes, but it also suffers from the underfitting problem which damages robustness. Especially, there are two problems which could be called sparse context and obsolete context problems. A sparse context means insufficient training data leading to underfitting. An obsolete context corresponds to a different task distribution problem between past and current time steps due to task shift.

To handle two problems, (Yoon et al., 2020) suggensts a **Recurrent Memory Recounstruction** (**RMR**) mechanism which generates a imaginary context $\tilde{C}_t$ for sparsity and reforms it through a recurrent updating module.

The RMR wants to generate imaginary keys set $\tilde{X}_t = \{\tilde{x}_t^{(k)}\}_k$ for applying proper attentions according to a task-shift first. Here, $k \in [1, K]$ which means the imaginary context with K memory cells (Hyperparameter). We are given a real context C_t, previous keys $\tilde{X}_{t-1}$ and previous RNN hidden state $\tilde{h}_{t-1}^{key}$. The key-generation process is:

$$(\tilde{X}_t, \tilde{h}_t^{key}) = GenerateKey(C_t, \tilde{X}_{t-1}, \tilde{h}_{t-1}^{key})$$

Then, it generates imaginary values set $\tilde{V}_t = \{\tilde{v}_t^{(k)}\}_k$.

$$\tilde{V}_t, \tilde{H}_t^{val} = GenerateValue(\tilde{X}_t, C_t, \tilde{V}_{t-1}, \tilde{H}_{t-1}^{val})$$

where $\tilde{H}_{t-1}^{val}$ is a set of hidden states for value generation.

Value generation module consists of tracking and interaction parts, where RNN and self-attention are applied.

Now, we formulate the whole RMR. The main task of RMR is to generate a imaginary context $\tilde{C}_t$ for each step $t \in [1, T]$ when we are given a current real context C_t and previous key-value pairs $(\tilde{x}_t^{(k)}, \tilde{y}_t^{(k)})$ for $k \in [1, K]$. Let $\tilde{\mathcal{H}}_t = H_t^{key} \cup H_t^{val}$ be the hidden states of RMR.

RMR is:

$$\tilde{C}_t, \tilde{\mathcal{H}}_t = RMR(C_t, \tilde{C}_{t-1}, \tilde{\mathcal{H}}_{t-1})$$

Since the RMR is just a memory, we should implement a reading operation. For each step t, we will conduct an attention on the combined context $\bar{C}_t = C_t \cup \tilde{C}_t$. Given a query x_t^q, a corresponding memory encoding is $r_{x_t^q} = Attention(\bar{C}_t; x_t^q)$.

We can construct a model called **Attentive Sequential Neural Process with RMR** (**ASNP-RMR**). The predictive distribution is:

$$P(Y, Z, \tilde{C}|X, C)$$
$$= \Pi_{t=1}^{T} P(y_t|x_t, z_t, \bar{C}_t) P(z_t|z_{<t}, \bar{C}_t) P(\tilde{C}_t|\tilde{C}_{<t}, C_{\leq t})$$

The third term is for updating a RMR.

Lastly, the ELBO of ASNP is:

$$\mathcal{L}_{ASNP} = \Sigma_{t=1}^{T} E_{Q_\phi(z_t|C,D)}[\log P_\theta(y_t|x_t, z_t, \tilde{C}_t, C_t)]$$
$$- E_{Q_\phi(z_{<t})}[KL(Q_\phi(z_t|z_{<t}, \tilde{C}_{\leq t}, C, D)|P_\theta(z_t|z_{<t}, \tilde{C}_{\leq t}, C_{\leq t}))]$$

4. Conclusion

In this project, we investigated Neural Process Family and its advanced version to improve. We checked the overall structure of NP briefly and delved into its mathematical foundation. Based on strong mathematical knowledge, we tracked the formulation processes of the individual models. Most of them are related to underfitting and robustness of NPs.

References

Garnelo, Rosenbaum, Maddison, Ramalho, Saxton, Shanahan, Teh, Rezende, and Eslami. Conditional neural processes. In *Proceedings of the 35th International Conference on Machine Learning (NeurIPS 2018)*, Stockholm, Sweden, 2018a.

Garnelo, Schwarz, Rosenbaum, Viola, Rezende, Eslami, and Teh. Neural processes. In *International Conference on Machine Learning (ICML 2018)*, 2018b.

Lee, Lee, Kim, Yang, Hwang, and Teh. Bootstrapping neural processes. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, Vancouver, Canada, 2020.

Singh, Yoon, Son, and Ahn. Sequential neural processes. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada, 2019.

Yoon, Singh, and Ahn. Robustifying sequential neural processes. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, Vancouver, Canada, 2020.