A Review on Neural Processes

Youngbin Lee[1]

Abstract

With the need of a novel and competitive prediction model, Neural Processes (NPs) have been introduced. NPs try to improve both Gaussian Processes (GPs) and Neural Networks (NNs) by learning a prior about the distribution of ground truth functions from data on its own. Due to its important application value, there have been emerging NP-related works. In this paper, the main ideas about general algorithms of NPs are first described. Then, the follow-up studies are reviewed with their contributions. Moreover, we analyze the challenges of applying NPs on different types of data and discuss how existing works address these challenges.

1. Introduction

Deep neural networks (DNNs) have been dominating numerous tasks in pattern recognition and machine learning (1). With its computational efficiency for training, a deep supervised learning having multiple layers has turned out to be useful in many different areas such as image recognition and natural language processing.

However, DNNs have limitations that it cannot make predictions that are flexible with regards to the size and order of inputs. In other words, output is no longer updated once the model has done with training. The reason being is that DNNs as a regression task are cast as modelling a deterministic function (i.e., It approximates only a single function). This can be addressed by another way of regression task, which is modelling distribution over functions. In this way, the model can not only adapt to new data but also estimate the uncertainty about predictions.

Thus, there can be a better regression model that combines a neural network and inference on distribution over functions, so-called stochastic processes. This concept was introduced as the name of Neural Processes (NPs) (2) and there have been follow-up methods making contributions to its algorithms and applications.

There are two main sub-families (i.e., Collection of models) that share the main algorithm of NP, according to the way to model the predictive distribution. One is conditional NP family where the predictive distributions of target outputs are consistent with each other for different target inputs. We call this factorisation assumption because the predictive distribution is factorised conditioned on the context set. In this sub-family, examples include Conditional Neural Processes (CNP) (3), Attentive Conditional Neural Process (ACNP), Convolutional Conditional Neural Process (ConvCNP), etc. The other sub-family is latent NP family where the predictive distributions of target outputs include a latent variable z. This allows the models to be applicable to new tasks such as producing coherent samples from the same target input. In this sub-family, examples include Neural Processes (NP), Attentive Neural Processes (ANP) (4), Convolutional NP (ConvNP), etc.

In this paper, we first introduce basic concepts to understand NP in section 2. Then, the main algorithms and follow-up models are reviewed with their experimental results in section 3. Finally, we conclude by discussing how the variants of NPs handle challenges of NPs in section 4.

2. Background

There are a variety of approaches that stem from NP and most of them share the same concepts and terms such as assumptions and training of model. To train NP, we need a dataset consisting of functions $f : X \to Y$ which are sampled from some distribution D. For example, for a function $f_d(x)$ the dataset is tuples $(x, y)_i$ where $y_i = f_d(x_i)$. And we divided this dataset into context $C = \{(x, y)_i\}_{i=1}^{n}$ and target $T = \{(x, y)_i\}_{i=1}^{n+m}$ with m additional unobserved points. At test time, the model should predict $y_T = f(x_T)$ with considering C.

2.1. Stochastic Process

A stochastic process P defines the probability distribution over functions f (i.e., We can sample f from P). Considering a task for predicting $f(x_i)$ for every target given observations, P defines a conditional distribution of $f(x_i)$ given

[1]Department of Industrial Engineering, UNIST. Correspondence to: Youngbin Lee <young@unist.ac.kr>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

target and observation. With stochastic processes, we can continuously estimate predictive distribution by Bayesian inference and it leads to a data efficiency. However, in this case, we need a prior to f and this is defined by kernel function (e.g., pairwise correlation structure of data) when we use GPs.

2.2. Meta-learning

Neural Networks can approximate only a single function to a ground truth function. And this leads to a poor and unreliable predictions when we don't have much data to train. Meta-learning tries to incorporate data from many related tasks and make model flexible with regard to input size and order. They are often used for few-shot estimations and this is where meta-learning and NPs are related. NPs themselves are models that do few-shot learning because they try to make predictions (or estimate a function) by looking at the predictive distribution conditioning on input-output pairs that are drawn from the function at test time.

3. Models

3.1. Conditional Neural Processes (CNP)

The original NPs are introduced with two different versions: Conditional NPs (CNPs) and NPs. We first review the former because its architecture is simpler, which leads to an easier understanding of the concept of NPs. To begin with, we define an observation set O as pairs of inputs and outputs, and a target set T as unlabelled input points.

As stated in section 1, NPs try to find distribution over functions. And particularly, CNPs define conditional distribution over functions given observations. In other words, the model depends on observation and it is where neural network kicks in. The dependency on observation is parametrized via embedding with neural networks. Formally, each observation (input-output pair) is first converted into representations r_i with an encoder h_θ.

$$r_i = h_\theta(x_i, y_i) \quad \forall (x_i, y_i) \in O \tag{1}$$

Then, the representations are aggregated to a single representation r with maintaining the same dimension.

$$r = r_1 + ... + r_n \tag{2}$$

Finally, the single representation r containing information about observations goes to decoder g_θ when making predictions of each target inputs.

$$\phi_i = g_\theta(x_i, r) \quad \forall (x_i) \in T \tag{3}$$

Assuming a factored structure, ϕ_i here indicates parameters of an output that the model produces. In regression task, ϕ_i contains mean and variance because it parametrizes Gaussian distribution about target inputs. Thus, we can say that CNPs make use of prior about functions to estimate new function with regard to test data like Bayesian approaches. However, unlike any other Bayesian regression models like GPs where humans degin an appropriate prior, CNPs learns prior from train data by itself. This is why CNPs are scalable and flexible at test time.

Training CNPs, a conditional stochasfic process Q_θ, is conducted to make model predict O using a random subset of O. That is, minimizing the negative conditional log probability

$$L(\theta) = -\mathbb{E}_{f\sim P}[\mathbb{E}_N[logQ_\theta(\{y_i\}_{i=0}^{n-1}|O_N, \{x_i\}_{i=0}^{n-1})]] \tag{4}$$

The performance of few-shot learning is demonstrated in 1D regression task, as CNPs learn better than GPs with a few data (5 context points). Image completion is another task that we can see the model's capability to predict under context. With CelebA dataset, CNPs showed flexible predictions even when conditioned on a subset of data that they did not see in training. This cannot be captured in GPs and we cancsay that CNPs can extract prior knowledge from train data.

3.2. Neural Processes (NP)

Since CNPs assumed a factored structure that only produces predictive mean and variance, they cannot make coherent samples on the same input whereas GP can because we can draw samples from multivariate Gaussian where the model captures covariance between every data point. This is important because one of the main reasons to model distribution of functions is to estimate uncertainty in predictions. NPs are able to produce samples of functions from the same context points by introducing latent variable z. The architecture of NPs is similar to that of CNPs except the latent variable z that captures a global uncertainty. That is, in NPs, a Gaissian distribution of the latent variable z is parametrized by the single global representation of context data and this latent variable z is used to make predictions by putting into decoder g. In other words, we can sample z from r whenever we want to produce a coherent sample from some target input.

Formally, NPs parametrize a stochastic process F with a high-dimensional random vector z and use some learnable function g.

$$F(x) = g(x, z) \tag{5}$$

Assuming the latent distribution $p(z)$ is a multivariate standard normal, the evidence lower-bound (ELBO) to be used

to train is derived as the following:

$$log\ p(y_{m+1:n}|x_{1:n}, y_{1:m}) \geq \mathbb{E}_{q(z|x_{1:n},y_{1:n})}[\Sigma_{i=m+1}^{n} log\ p(y_i|z, x_i) + log\ \frac{p(z|x_{1:m}, y_{1:m})}{q(z|x_{1:n}, y_{1:n})}]$$

What NPs are distinct from other models that extract representations from data using neural network such as Matching Networks and Depp kernel learning is that it learns a data-driven prior and this leads to a computational efficiency. Experimental results show that NPs take four times fewer iterations than random search on Bayesian optimization using Thompson sampling and NPs produce coherent images which are the important characteristics of the model.

3.3. Attentive Neural Processes (ANP)

Although (C)NPs successfully combined the strengths of both GPs and NNs, it suffers from the problem of underfitting. In other words, it cannot predict well on the context dataset. This is because the mean-aggregation of encoder gives the same weights to every context point. When the decoder makes predictions, there must be certain context points that are useful and relevant. For this reason, ANPs try to learn to distinguish relevent context points by measuring similarities between context points and query point using attention.

The architecture of the model is modified from original NPs as follows: Instead of an encoder, a self-attention is applied when calculating representations from context points to capture interactions between them. Then, a cross-attention is used to aggregate representations to make the model attend to the most relevant context. The decoder is the same as NPs and we can say that ANPs are a generalization of NPs because it becomes NPs when using a uniform attention that gives the same weights to every context point.

Because attention mechanism is added to original NPs, the computational complexity is raised from $O(n + m)$ to $O(n(n + m))$. However, despite of an increased prediction time, an experimental result on training loss showed that ANPs are trained faster because attention mechanism is calculated in parallel.

A quantitative experimental results on 1D function regression showed that the loss and training time decreased in ANPs compared to NPs. A qualitative results with a graph of predictive mean and variance show that the underfitting of original NPs is resolved. In 2D function regression task, it is notable that ANPs showed a flexibility to perform mapping images to higher resolution that original NPs cannot do. A multi-head attention made it possible and more application of ANPs is expected if attention mechanism is applied in decoder.

3.4. Sequence Neural Processes (SNP)

Another area to be developed from original NPs is a dynamically changing sequence of stochastic processes. For example, the Generative Query Networks (GQN) which is a variant of NPs were able to model static 3D scenes but cannot handle dynamic 3D scenes. SNPs (5), on the other hand, incorporated temporal transition model into NPs to handle dynamic scenes.

The generative process of SNPs includes temporal structure and this is a Recurrent State-Space Model (RSSM) version of SNPs.

$$P(Y, Z|X, C) = \Pi_{t=1}^{T} P(Y_t|X_t, z_t)P(z_t|z_{<t}, C_t) \quad (6)$$

In the paper, GQN is extended to a Temporal GQN (TGQN) using SNPs. The purpose of this model is to transfer knowledge from previous scene in a dynamic 3D modelling. TGQN demonstration in experiments showed that it better transfer past knowledge than GQN as the negative log-likelohood decreased.

Another novel part of SNPs is the use of state transition $P(z_t|z_{<t}, C_t)$ that is conditioned on the context, which enables a meta-transfer learning. However, the model tends to ignore C_t (i.e., transition collapse problem) and the paper suggests posterior dropout to solve this problem. Its design of the approximate posterior of Z and corresponding ELBO is to limit information contained in $z_{<t}$ and encourage the use of C_t.

3.5. Functional Neural Processes (FNP)

Latent NPs need to posit a latent prior distribution $p(z)$ to train the model. However, it can make the model complicated and FNPs (6) are designed to be simpler by learning a graph instead of an explicit global latent variable. Here, the graph indicates dependency structure on the embedding space of data points.

Holding the two conditions of exchangeability and consistency, FNPs try to encode prior assumptions and inductive biases to the model instead of specifying a latent prior.

FNPs build a graph of dependencies between input embeddings. With a reference set $R = \{x_1^r, ..., x_K^r\}$ that is selectied from X and the other set $O = X \setminus R$, let B is the union of training points and R. Then, the dependency graph is constructed using correlations between points in B. This acts like a kernel function in GPs. A directed acyclic graph G is built with the points in R and a bipartite graph A is built from R to M. Here, M is $D_x \setminus R$ where D_x is any finite random set from X. Then, the predictive distribution depends on the reference set$R$ according to graphs G and A.

Experimental results compare FNPs with other models such

as NNs and NPs. The inductive bias that is encoded in FNPs is verified with visualization with predictive distributions in 1D regression function. Also, FNPs outperform other competitive models including NPs in prediction accuracy. Still, FNPs can be improved with better function priors. For example, one could use a manifold structure on the latents.

4. Conclusion

In this review, the main ideas of NPs are presented with its variants. We can use NPs to model distributions over functions that are flexible for high dimensional problems, with faster training and inference. However, we found limitations and uncovered areas of NPs and other follow-up methogologies to handle those problems. Attentive NPs try to solve underfitting of context set and to be flexible to new applications in image like mapping resolutions. Sequence NPs try to better temporal modelling and transfer of past knowledge, which leads to 3D modelling in dynamic scenes. FNPs try to make NPs simpler by using a graph of dependencies on top of latent representations of data points instead of using global latent variables.

References

[1] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural networks, 61, 85-117.

[2] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh. Neural processes. In arXiv:1807.01622, 2018.

[3] M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. M. A. Eslami. Conditional neural processes. In arXiv:1807.01613, 2018.

[4] Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., ... Teh, Y. W. (2019). Attentive neural processes. arXiv preprint arXiv:1901.05761.

[5] Singh, G., Yoon, J., Son, Y., Ahn, S. (2019). Sequential neural processes. Advances in Neural Information Processing Systems, 32.

[6] Louizos, C., Shi, X., Schutte, K., Welling, M. (2019). The functional neural process. Advances in Neural Information Processing Systems, 32.