Stick-Breaking Neural Latent Variable Models

Daniel Flam-Shepherd, Yuxiang Gao, Zhaoyu Guo
University of Toronto
{danielfs, ygao, kenny}@utstat.toronto.edu

Abstract

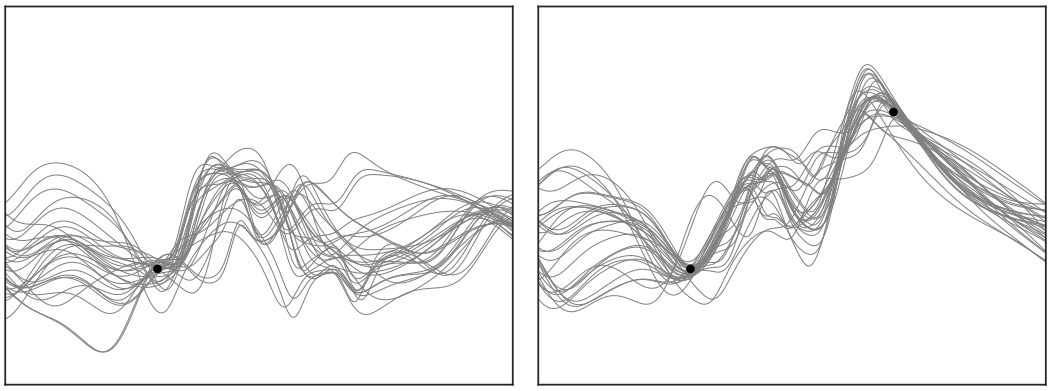
Neural processes [1] define a class of *neural latent variable models*. We extend this class to an infinite dimensional space by imposing a stick-breaking prior on the latent space. Using Stochastic Gradient Variational Bayes, we perform posterior inference for the weights of the stick-breaking process and develop the stick-breaking neural process (SB-NP). SB-NPs are able to learn the dimensionality of the latent space and have improved posterior uncertainty.

1 Introduction and Motivation: Neural Latent Variable Models

For any neural network-based model that uses latent variables $\mathbf{z}$ and data $\mathbf{x}$ as inputs : $f(\mathbf{x}, \mathbf{z})$ can be considered a *neural latent variable model*. For example, Bayesian neural networks (BNN) [2] have recently been extended with latent variables (LVBNN) [3] to model complex stochastic functions. In this work, we'll consider the recently proposed neural processes.

NPs are distributions over functions consisting of an encoder-decoder architecture similar to the variational autoencoder [4], where functional uncertainty is driven by a global latent variable $\mathbf{z}$ with fixed dimensionality. However, neural processes have poor uncertainty quality which often collapses to a point estimate. To get reasonable uncertainty quality, it requires careful tuning of model architecture and latent dimensionality.

In this work, we extend NPs by using an infinite dimensional latent variable through the use of a stick-breaking prior, so that the posterior can find the dimensions that are significant to function uncertainty. Our model, the stick-breaking neural process (SB-NP), *learns* stick-breaking weights for every latent dimension so that its uncertainty in functions does not collapse during training.



(a) one data point (b) two data points

Figure 1: Plots of functions sampled from the SB-NP posterior, conditioning on one and two data points.

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

2 Background Information

2.1 Stick-breaking process

A random measure is referred to as a stick-breaking prior (SBP) [5] if it is of the form $G(\cdot) = \sum_{k=1}^{\infty} \pi_k \mu_{\delta_k}$ where μ_{δ_k} is a dirac measure on $\delta_k \sim G_0$ and G_0 is a base distribution [5]. The π_k are random weights independent of G_0, chosen such that $0 \leq \pi_k \leq 1$ and $\sum_k \pi_k = 1$ almost surely. The weights can be drawn according to the following iterative procedure for $k \geq 1$:

$$v_k \sim \text{Beta}(\alpha, \beta), \quad \pi_1 = v_1, \quad \pi_k = v_k \prod_{j<k} (1 - v_j) \tag{1}$$

When $v_k \sim \text{Beta}(1, \alpha_0)$ this is denoted as $\boldsymbol{\pi} \sim \text{GEM}(\alpha_0)$ [6]. One can obtain stochastic gradients with respect to the parameters of the Beta distribution using implicit or general reparameterization gradients [7]. However, we can use a reparameterizable proxy instead, similar to [8].

2.2 The Kumaraswamy distribution

The Kumaraswamy distribution [9] is used in our SB-NP model, as it admits reparameterization gradients that are easy to compute numerically. It closely resembles the Beta distribution and admits reparameterizable samples via its closed-form inverse CDF. It has the following density

$$p(x|a, b) = abx^{a-1}(1 - x^a)^{b-1} \text{ for } x \in (0, 1) \text{ and } a, b > 0 \tag{2}$$

Furthermore, the KL divergence between the Kumaraswamy and the Beta distribution has a closed-form. Let $p(v_k)$ have the distribution $\text{Beta}(\alpha, \beta)$ and let $q_{\boldsymbol{\phi}}(v_k)$ have the distribution $\text{Kumaraswamy}(a_{\boldsymbol{\phi}}, b_{\boldsymbol{\phi}})$. Then the KL divergence is

$$\begin{aligned} \mathbb{KL}[q_{\boldsymbol{\phi}}(v_k)||p(v_k)] = \frac{a_{\boldsymbol{\phi}} - \alpha}{a_{\boldsymbol{\phi}}} \left[\gamma - \Psi(b_{\boldsymbol{\phi}}) - \frac{1}{b_{\boldsymbol{\phi}}} \right] + \log a_{\boldsymbol{\phi}} b_{\boldsymbol{\phi}} + \log B(\alpha, \beta) \\ + (\beta - 1) b_{\boldsymbol{\phi}} \sum_{m=1}^{\infty} \frac{1}{m + a_{\boldsymbol{\phi}} b_{\boldsymbol{\phi}}} B\left(\frac{m}{a_{\boldsymbol{\phi}}}, b_{\boldsymbol{\phi}} \right) - \frac{b_{\boldsymbol{\phi}} - 1}{b_{\boldsymbol{\phi}}} \end{aligned} \tag{3}$$

where γ is Euler's constant, Ψ is the digamma function, $B(a, b)$ is the beta function. Note that the RHS of the above equation has no dependency on the index k. This KL divergence contains an infinite sum but this can be approximated accurately with the first few terms [8].

2.3 Neural Processes

NPs, like Gaussian Processes, are able to adapt to new observations and learn complex distributions of functions. In NPs, the training data $\mathcal{D} = \mathcal{D}_t \cup \mathcal{D}_c$ is divided into a target set $\mathcal{D}_t$ and context set $\mathcal{D}_c$, with both sets being disjoint. A variational approximation $q(\mathbf{z}|\mathcal{D})$ to true posterior $p(\mathbf{z}|\mathcal{D})$ is learned by maximizing the conditional evidence lower bound:

$$\begin{aligned} \log p(\mathcal{D}_t|\mathcal{D}_c) &= \log \mathbb{E}_{q(\mathbf{z}|\mathcal{D})} \left[\frac{p(\mathcal{D}_t, \mathbf{z}|\mathcal{D}_c)}{q(\mathbf{z}|\mathcal{D})} \right] \geq \mathbb{E}_{q(\mathbf{z}|\mathcal{D})} \left[\log \frac{p(\mathcal{D}_t, \mathbf{z}|\mathcal{D}_c)}{q(\mathbf{z}|\mathcal{D})} \right] \\ &\geq \mathbb{E}_{q(\mathbf{z}|\mathcal{D})} [\log p(\mathcal{D}_t|\mathcal{D}_c, \mathbf{z}) + \log q(\mathbf{z}|\mathcal{D}_c) - \log q(\mathbf{z}|\mathcal{D})] \end{aligned} \tag{4}$$

where we have approximated the conditional prior with the approximate posterior $q(\mathbf{z}|\mathcal{D}_c) \approx p(\mathbf{z}|\mathcal{D}_c)$. Neural processes include:

1. an encoder network $q(\mathbf{z}|\mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}}(\mathcal{D}), \boldsymbol{\sigma}_{\mathbf{z}}(\mathcal{D}))$ where $\boldsymbol{\mu}_{\mathbf{z}}(\mathcal{D}), \boldsymbol{\sigma}_{\mathbf{z}}(\mathcal{D}) = \mathbf{a}(\boldsymbol{h}_{\boldsymbol{\phi}_h}(\mathcal{D}))$ with parameters $\boldsymbol{\phi}_h$.
2. a decoder network $\boldsymbol{f}_{\boldsymbol{\phi}_f}(\mathbf{x}, \mathbf{z})$ that makes predictions with parameters $\boldsymbol{\phi}_f$.
3. an aggregator $\mathbf{a}(\mathbf{r}) = \mathbb{E}_n(\mathbf{r})$ which takes the mean over the data in the latent space, with $r_i = \boldsymbol{h}_{\boldsymbol{\phi}_h}((\mathbf{x}_i, y_i))$ and $\mathbb{E}_n$ being the empirical average.

Algorithm 1 describes the training procedure of the SB-NP. $\boldsymbol{h}$ is the encoder network and $\boldsymbol{f}$ is the decoder network. ϕ_h, ϕ_f are the parameters of the encoder and decoder networks, respectively. adam() denotes use of the adam optimization algorithm [10]

Initialize $\phi = \{\phi_{\boldsymbol{h}}, \phi_{\boldsymbol{f}}\}$
while ϕ not converged **do**
  $\mathbf{r} = \boldsymbol{h}_{\phi_h}(\mathbf{X}, \mathbf{y})$ ▷ output from the encoder
  $\mathbf{a}, \mathbf{b} = \exp(\mathbb{E}_n[\mathbf{r}_n])$ ▷ aggregate over the data to get the parameters
  $\mathbf{u} \sim U(0, 1)$ ▷ sample from the uniform
  $\mathbf{v} = (1 - \mathbf{u}^{1/\mathbf{a}})^{1/\mathbf{b}}$ ▷ sample from the Kumaraswamy distribution
  $\boldsymbol{\pi} = (v_1, v_2(1 - v_1), \dots)$ ▷ get the weights of stick-breaking process
  $\hat{\boldsymbol{y}} = \boldsymbol{f}_{\phi_f}(\mathbf{X}, \boldsymbol{\pi})$ ▷ use the decoder to make a prediction
  $\mathbf{g}_\phi \leftarrow \hat{\nabla}_\phi \mathcal{L}_\mathcal{D}(\phi)$ ▷ estimate gradients of the elbo
  $\phi \leftarrow \text{adam}(\phi, \mathbf{g}_\phi)$ ▷ update parameters
Return ϕ^*

3 Stick-breaking neural processes (SB-NP)

We incorporate the use of a stick-breaking prior into the neural process architecture. The generative model of SB-NP is identical to NPs except that the latent variables $\boldsymbol{\pi}$ are drawn from the stochastic process GEM(α_0). This is similar in nature to the stick-breaking variational autoencoder [8]. The generative model can be described simply as

$$\boldsymbol{\pi} \sim \text{GEM}(\alpha_0) \quad \mathbf{y}|\boldsymbol{\pi} \sim p(\mathcal{D}|\boldsymbol{\pi})$$

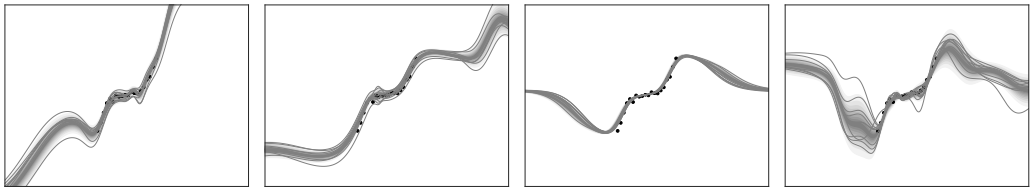
We use Stochastic Gradient Variational Bayes for posterior inference on the weights of a stick-breaking process. For inference, the encoder outputs the parameters of the Kumaraswamy distribution which are used for sampling v_k. A truncation level of K is set and then the stick segments are composed as $\boldsymbol{\pi} = (v_1, v_2(1 - v_1), \dots, \prod_{j \leq K}(1 - v_j))$. In contrast to NP, we can typically select it to be fairly small ($k < 10$), in order to obtain meaningful uncertainty. The training procedure of SB-NPs is described in Algorithm 1. To use a tractable prior, unlike in neural processes, we do not split the data into a context and target set and optimize the following lower bound:

$$\begin{aligned} \log p(\mathcal{D}) &= \log \mathbb{E}_{q(\boldsymbol{\pi}|\mathcal{D})}\left[\frac{p(\mathcal{D}, \boldsymbol{\pi})}{q(\boldsymbol{\pi}|\mathcal{D})}\right] \geq \mathbb{E}_{q(\boldsymbol{\pi}|\mathcal{D})}\left[\log \frac{p(\mathcal{D}, \boldsymbol{\pi})}{q(\boldsymbol{\pi}|\mathcal{D})}\right] \\ &= \mathbb{E}_{q(\boldsymbol{\pi}|\mathcal{D})}[\log p(\mathcal{D}|\boldsymbol{\pi})] - \mathbb{KL}[q(\boldsymbol{\pi}|\mathcal{D})||p(\boldsymbol{\pi})] := \mathcal{L}_\mathcal{D}(\phi) \end{aligned} \tag{5}$$

Using the Kumaraswamy distribution allows us to take stochastic gradients of the objective using the reparameterization trick. Samples can be drawn via the inverse transform $x = (1 - u^{\frac{1}{b}})^{\frac{1}{a}}$ where $u \sim \text{Uniform}(0, 1)$. We utilize `autograd` [11] to calculate gradients of the lower bound by back-propagating through the variational posterior sampling procedure. We use the adam optimization algorithm [10] to update the parameters.

(a) NP (b) SB-NP

Figure 2: On the left, (a) is the graphical model of a NP and b) the graphical model of a SB-NP.



(a) BNN (b) LVBNN (c) NP (d) **SB-NP**

Figure 3: Plots of the 4 posteriors for data sampled from a cubic function. Different shades of color correspond to deciles of the predictive density. Note that SB-NP has better posterior uncertainty.

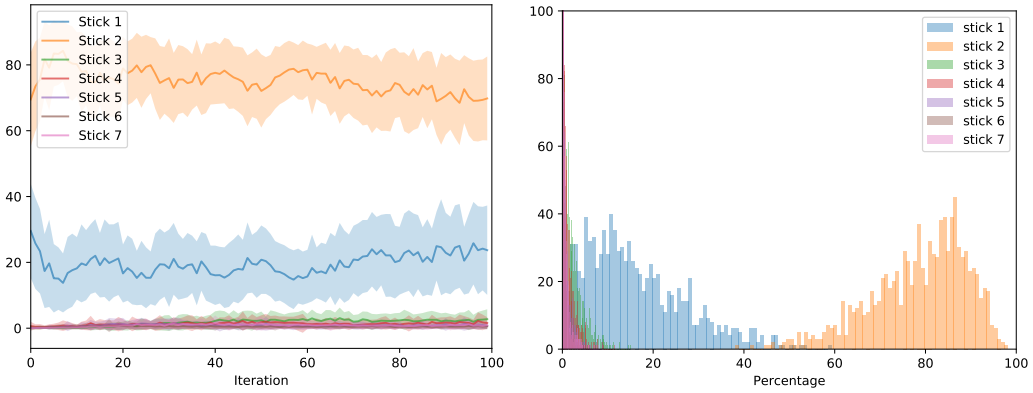
4 Results on toy data

We compare our stick-breaking neural latent variable model (SB-NP) with a bayesian neural net, a latent variable bayesian neural net and a neural process. Figure 3 displays the posteriors on toy problem $y = f(x) + \epsilon$ where $f(x) = x^3$ and ϵ is gaussian noise.

While uncertainty in neural processes collapses, the SB-NPs uncertainty doesn't collapse during training. Furthermore the SB-NP has larger uncertainty bands away from the data than both the BNN and LVBNN. This demonstrates that stochastic dimensionality in the latent space gives rise to better posterior uncertainty.

During training on another toy problem we learn a 7 dimensional posterior on π. We visualize the latent space in Figure 4, where a) displays the convergence of the sticks during training and b) displays the distribution of sticks after training.

According to Figure 4: Stick weights π_1, π_2 on average have the top weighting at the end of training and are the main factors driving posterior uncertainty. The SB-NP learns a lower dimensional latent space than a neural process, which usually must be hand tuned to a fixed high dimensional latent space.



(a) Average stick weights with σ-bands (b) Histogram of sticks

Figure 4: Behaviour of stick breaking weights during training

5 Conclusion and Future Work

In this work, we proposed the stick-breaking neural latent variable models (SB-NP), which are distributions of functions that use a global latent variable of stochastic dimensionality. We demonstrate the expressiveness of SB-NPs in modelling posterior uncertainty in comparison to bayesian neural nets and neural processes. Placing priors on the decoder network and training SB-NPs on much larger datasets are future directions of this work.

6 Acknowledgements

We thank Yanbo Tang and Tommy Guo for the helpful feedback and discussions.

References

[1] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S.M. Ali Eslami, and Yee Whye Teh. Neural processes. *arxiv preprint*, 2018.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *International Conference on Machine Learning*, 2015.

[3] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. *International Conference on Machine Learning*, 2018.

[4] D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *International Conference on Learning Representations*, 2014.

[5] Hemant Ishwaran and Lancelot F James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 2001.

[6] Jim Pitman. Combinatorial stochastic processes. *UC Berkeley Technical Report*, 2002.

[7] Michael Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit reparameterization gradients. *arxiv preprint*, 2018.

[8] Eric Nalisnick and Padhraic Smyth. Stick breaking variational autoencoders. *ICLR*, 2017.

[9] Ponnambalam Kumaraswamy. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 1980.

[10] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 2015.

[11] Dougal Maclaurin, David Duvenaud, Matthew Johnson, and Ryan P. Adams. Autograd: Reverse-mode differentiation of native python. 2015.