Mapping Gaussian Process Priors to Bayesian Neural Networks

Daniel Flam-Shepherd
University of Toronto

James Requeima
University of Cambridge

David Duvenaud
University of Toronto

Abstract

This paper describes a method to "map" a Gaussian process prior to a Bayesian Neural Net. This is done by minimizing over some data distribution of interest the kullback leibler divergence from prior distributions of the Gaussian process to the Bayesian Neural Net (in function space). This allows of to train a Bayesian Neural Net with apriori knowledge of the covariance structure of the data distribution in the input space.

1 Introduction and Motivation

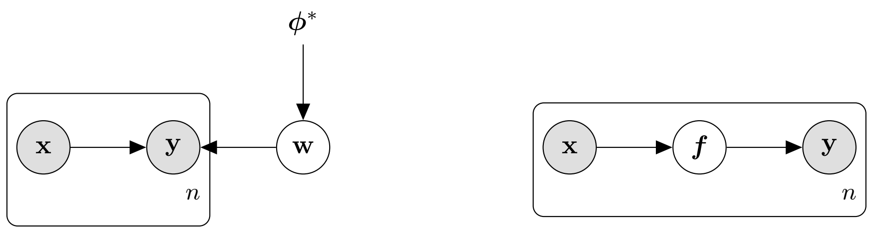
What defines a reasonable prior to use when forming and training Bayesian models and Bayesian neural networks? In a recent work, (Ghosh et al 2016) apply a horseshoe prior over preactivations of a Bayesian neural network to effectively turns off weights that do not help explain the data. That model and many Bayesian models view priors solely in parameter space, in this work we move forward with viewing priors in function space as well.

It is difficult to incorporate meaningful prior information about functions to be modelled by BNNs since priors are generally specified over the network parameters. Often, normal distributions are placed over the weights for convenience and are interpreted as a bias toward less complex functions via smaller weights. Gaussian processes, on the other hand, have a elegant mechanism for incorporating prior beliefs about the underlying function - specifying the mean and covariance functions. However, Gaussian Process have scalability limitations making bayesian neural networks a more practical model in large data settings. In our work, we present an approach to specify a more principled prior for Bayesian Neural Networks that can leverage the well studied kernel design techniques from Gaussian process regression.

We consider matching the prior of a Bayesian neural network $p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})$ to the prior of a Gaussian process $p_{\text{GP}}(\boldsymbol{f}|\boldsymbol{\theta})$ by minimizing their KL divergence over some data distribution of interest $\mathbf{X} \sim p(\mathbf{X})$. We minimize the KL divergence with respect to the initial variational parameters $\boldsymbol{\phi}$ of the proposal distribution $q(\mathbf{w}|\boldsymbol{\phi})$. These variational parameters $\boldsymbol{\phi}^* = \{\boldsymbol{\mu}^*_{\boldsymbol{\phi}}, \log \boldsymbol{\sigma}^*_{\boldsymbol{\phi}}\}$ yield a prior on the BNN weights $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}^*_{\boldsymbol{\phi}}, \boldsymbol{\sigma}^*_{\boldsymbol{\phi}}) \equiv q(\mathbf{w}|\boldsymbol{\phi}^*)$. Then, variational inference allows us to perform approximate inference in our BNN using this more principled prior. In both stochastic optimization steps, we use the reparameterization trick **?**] to sample from the weights $\mathbf{w}$ and draw functions from our BNN. We describe the implementation of both steps in the next section.

1.1 Contributions

- We formulate two methods to map the prior of a Gaussian process to a Bayesian neural network working strictly in function space. The first minimizes the distributions over



(a) BNN (b) GP

Figure 1: (a) and (b) display the graphical models of a BNN and GP.

functions of the BNN to the GP. The second trains in adversarial fashion a BNN prior to generate functions from a data distribution that resemble functions sampled from a GP prior.

- We provide an stochastic optimization algorithm for the implementation of both methods using the reparametrization trick.
- We describe how to do kernel hyperparameter optimization within our formalism using continuous methods that directly and indirectly optimize the log marginal likelihood (TO DO)
- We test our methods in a variety of toy regression tasks using synthetic data and (TO DO) some selected datasets from the UCI repository.

2 Background on BNNs and GPs

2.1 Bayesian Neural Networks

To do inference in BNN model we learn a variational approximation $q(\mathbf{w}|\boldsymbol{\varphi})$ to $p(\mathbf{w}|\mathcal{D})$ the posterior distribution of the parameters $\mathbf{w}$ conditioned on training data $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$ where $p(\mathbf{w}|\mathcal{D}) = p(\mathcal{D}|\mathbf{w})p(\mathbf{w})/p(\mathcal{D})$ where $p(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})d\mathbf{w}$ Learning the posterior on the parameters will mean updating our prior from $\mathbf{w} \sim p(\mathbf{w})$ to $\mathbf{w} \sim p(\mathbf{w}|\mathcal{D})$ as a consequence of the evidence from the training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N \sim p(\mathcal{D})$. This distribution $p(\mathbf{w}|\mathcal{D})$ allows us to form a predictive distribution by integrating out $\mathbf{w}$.

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \int p(\mathbf{y}|\mathbf{x}, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w} = \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[p(\mathbf{y}|\mathbf{x}, \mathbf{w})]$$

In SVI, we use the reparametrization trick to obtain a stochastic optimization problem on the parameters $\boldsymbol{\phi}$ of $q_{\boldsymbol{\phi}}(\mathbf{w})$ through minimizing the KL divergence $\mathbb{KL}[q_{\boldsymbol{\phi}}(\mathbf{w})|p(\mathbf{w}|\mathcal{D})]$ or maximizing the evidence lower bound (ELBO) $\mathcal{L}(\boldsymbol{\phi})$ of the marginal likelihood of the data:

$$\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi}}{\text{argmin}}\, \mathbb{KL}[q_{\boldsymbol{\phi}}(\mathbf{w})|p(\mathbf{w}|\mathcal{D})] = \underset{\boldsymbol{\phi}}{\text{argmax}}\, \mathcal{L}_{\mathcal{D}}(\boldsymbol{\phi})$$

where $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}[\log p(\mathcal{D}|\mathbf{w}] - \mathbb{KL}[q_{\boldsymbol{\phi}}(\mathbf{w})|p(\mathbf{w})]$

2.2 Gaussian Processes

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution. A Gaussian process is completely specified by its mean and covariance function; $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \text{ and } k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{1}$$

choose some subset of inputs $\mathbf{X}_\bullet = \{\mathbf{x}_i\}$ and write $\mathbf{K}_{\bullet\bullet} = [k(\mathbf{x}, \mathbf{x}')]$ Then we generate a random Gaussian vector with this covariance matrix via $\boldsymbol{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\bullet\bullet})$ The joint prior distribution of the training outputs, $\boldsymbol{f} = f(\mathbf{X}_\bullet)$, and the test outputs $\boldsymbol{f}_* = f(\mathbf{X}_*)$ according to the prior is

$$(\boldsymbol{f}, \boldsymbol{f}_*) \sim \mathcal{N}(\mathbf{0}, \mathbf{C}), \;\; \mathbf{C} = \begin{pmatrix} \mathbf{K} & \mathbf{K}_{\bullet *} \\ \mathbf{K}_{*\bullet} & \mathbf{K}_{**} \end{pmatrix}$$

We can think of the formulation of GP as generating functions from the prior, and rejecting the ones that disagree with the observations. This corresponds to conditioning the joint Gaussian prior distribution on the observations .

For predicting noisy targets $\mathbf{y} = \boldsymbol{f} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma_n^2 \mathbb{I})$ where $\text{cov}(\mathbf{y}) = \mathbf{K} + \sigma_n^2 \mathbb{I} \equiv \mathbf{K}_y$ so that we can write the joint distribution of the observed target values and the function values at the test locations under the prior as $(\mathbf{y}, \boldsymbol{f}_*) \sim \mathcal{N}(\mathbf{0}, \mathbf{C}_*)$ where $\mathbf{C}_*$ replaces $\mathbf{K}$ with $\mathbf{K}_\mathbf{y}$. The predictive distribution is $p(\boldsymbol{f}_*|\mathbf{X}_*, \mathbf{y}, \mathbf{X}_\bullet) = \mathcal{N}(\mathbb{E}[\boldsymbol{f}_*], \text{cov}(\boldsymbol{f}_*))$ where

$$\mathbb{E}[\boldsymbol{f}_*] = \mathbf{K}_{\bullet *}\mathbf{K}_\mathbf{y}^{-1}\mathbf{y}, \;\text{ and }\; \text{cov}(\boldsymbol{f}_*) = \mathbf{K}_{**} - \mathbf{K}_{*\bullet}\mathbf{K}_\mathbf{y}^{-1}\mathbf{K}_{\bullet *}$$

3 Step One : Learning the prior parameters by Minimizing the KL

In this section we describe the procedure used to minimize the KL divergence of the BNN prior distribution over functions $p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})$ and the GP prior distribution over functions $p_{\text{GP}}(\boldsymbol{f}|\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ are hyperparameters of the kernel $\mathbf{K}$.

$$\mathcal{K}_{\mathbf{X}}(\boldsymbol{\phi}) = \mathbb{KL}[p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi}) \mid p_{\text{GP}}(\boldsymbol{f}|\boldsymbol{\theta})] = \int p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi}) \log \left[\frac{p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})}{p_{\text{GP}}(\boldsymbol{f}|\boldsymbol{\theta})}\right] d\boldsymbol{f} \tag{2}$$

$$= -\mathbb{H}[p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})] - \mathbb{E}_{p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})}[\log p_{\text{GP}}(\boldsymbol{f}|\boldsymbol{\theta})] \propto -\frac{1}{S}\sum_{s=1}^{S} \log p_{\text{GP}}(\boldsymbol{f}^{(s)}|\boldsymbol{\theta}) \tag{3}$$

Where we have used a Monte Carlo estimate of $\mathbb{E}_{p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})}[\log p_{\text{GP}}(\boldsymbol{f}|\boldsymbol{\theta})]$, using S samples $\boldsymbol{f}^{(s)} \sim p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})$. We also assume that the entropy term $\mathbb{H}[p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})]$ is not a function of the variational parameters $\boldsymbol{\phi}$. We define our first stochastic optimization objective by approximating the KL divergence between these infinite dimensional distributions by taking expectations over where $p(\mathbf{X})$ allows us to prioritize where in the input space we want $p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi}) \sim p_{\text{GP}}(\boldsymbol{f}|\boldsymbol{\theta})$

$$\mathcal{L}_{\mathbf{X}}(\boldsymbol{\phi}) \equiv \mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})}[\mathcal{K}_{\mathbf{X}}(\boldsymbol{\phi})] \propto -\frac{1}{S}\sum_{s=1}^{S} \mathbb{E}_{p(\mathbf{X})}[\log p_{\text{GP}}(\boldsymbol{f}^{(s)}(\mathbf{x})|\boldsymbol{\theta})] \tag{4}$$

We minimize (3) $\boldsymbol{\phi}^* = \underset{\boldsymbol{\phi}}{\text{argmin}}\, \mathcal{L}_{\mathbf{X}}(\boldsymbol{\phi})$. This is described in Algorithm 1.

Algorithm 1 Learn the prior

```
1: Initialize $\boldsymbol{\phi} = \{\boldsymbol{\mu}_{\boldsymbol{\phi}}, \log \boldsymbol{\sigma}_{\boldsymbol{\phi}}\}$
2: while $\boldsymbol{\phi}$ not converged do
3:     $\boldsymbol{\epsilon}^{(s)} \sim p(\boldsymbol{\epsilon}) = \mathcal{N}(\mathbf{0}, \mathbb{I})$                  ▷ sample prior noise
4:     $\mathbf{w}^{(s)} \leftarrow \boldsymbol{g}(\boldsymbol{\phi}, \boldsymbol{\epsilon}^{(s)}) = \boldsymbol{\mu}_{\boldsymbol{\phi}} + \boldsymbol{\sigma}_{\boldsymbol{\phi}}\boldsymbol{\epsilon}^{(s)}$     ▷ sample $S$ weights $\mathbf{w} \sim q(\mathbf{w}|\phi)$
5:     $\mathbf{X} \leftarrow \{\mathbf{x}_1, \ldots, \mathbf{x}_n \sim p(\mathbf{x})\}$                     ▷ sample data
6:     $\boldsymbol{f}^{(s)} \leftarrow f(\mathbf{X}, \mathbf{w}^{(s)})$                     ▷ sample $S$ functions $\boldsymbol{f} \sim p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})$
7:     $\mathcal{L}_{\mathbf{X}}(\boldsymbol{\phi}) \leftarrow -\frac{1}{S}\sum_s \log p_{\text{GP}}(\boldsymbol{f}^{(s)}|\boldsymbol{\theta})$            ▷ compute the kl
8:     $\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\mathbf{X}}(\boldsymbol{\phi}) \leftarrow -\frac{1}{S}\sum_s \nabla_{\boldsymbol{\phi}} \log p_{\text{GP}}(\boldsymbol{f}^{(s)}|\boldsymbol{\theta})$   ▷ compute gradients of the kl
9:     $\boldsymbol{\phi} \leftarrow \text{adam}(\boldsymbol{\phi}, \nabla_{\boldsymbol{\phi}}\mathcal{L}_{\mathbf{X}}(\boldsymbol{\phi}))$              ▷ update the parameters
10: Return $\boldsymbol{\phi}^*$
```

4 Step One : Learning the prior parameters by an Adversarial Strategy

In this section we describe an adversarial procedure used to map the BNN prior distribution over functions $p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})$ to the the GP prior distribution over functions $p_{\text{GP}}(\boldsymbol{f})$. We treat the Bayesian neural network as the generator $G(\mathbf{X}, \mathbf{w}) = f_{\text{BNN}}(\mathbf{X}; \boldsymbol{t}(\boldsymbol{\phi}, \boldsymbol{\epsilon})) = \boldsymbol{f}_{\text{BNN}}$ and define a parametric neural network discriminator $D(\boldsymbol{f}, \boldsymbol{\theta}_d)$ that takes in samples of functions from the Gaussian process prior and Bayesian neural net 'generator' and uses parameters $\boldsymbol{\theta}_d$ to map them to a scalar : the probability that $\boldsymbol{f}$ came from $p_{\text{GP}}(\boldsymbol{f})$ rather than $p_{\text{BNN}}(\boldsymbol{f})$.

$$\mathcal{V}(\boldsymbol{\phi}_G, \boldsymbol{\theta}_D) = \mathbb{E}_{\boldsymbol{f} \sim p_{\text{GP}}(\boldsymbol{f})}\left[\log D(\boldsymbol{f})\right] + \mathbb{E}_{\boldsymbol{f} \sim p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})}\left[\log(1 - D(\boldsymbol{f}))\right] \tag{5}$$

$$= \int p_{\text{GP}}(\boldsymbol{f}) \log D(\boldsymbol{f}) d\boldsymbol{f} + \int p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi}) \log(1 - D(\boldsymbol{f})) d\boldsymbol{f} \tag{6}$$

$$\approx \frac{1}{S} \sum_{s=1}^{S} \log D(\boldsymbol{f}_{\text{GP}}^{(s)}) + \frac{1}{S} \sum_{s=1}^{S} \log(1 - D(\boldsymbol{f}_{\text{BNN}}^{(s)})) \tag{7}$$

Where we have used Monte Carlo estimates of $\mathbb{E}_{\boldsymbol{f} \sim p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi})}\left[\log(1 - D(\boldsymbol{f}))\right]$ and $\mathbb{E}_{\boldsymbol{f} \sim p_{\text{GP}}(\boldsymbol{f})}\left[\log D(\boldsymbol{f})\right]$ using S samples from $\boldsymbol{f}_{\text{BNN}}^{(s)} \sim p_{\text{BNN}}(\boldsymbol{f})$ and $\boldsymbol{f}_{\text{GP}}^{(s)} \sim p_{\text{GP}}(\boldsymbol{f})$

We define the stochastic optimization objective by approximating $\mathcal{V}_{\mathbf{X}}(\boldsymbol{\phi}_G, \boldsymbol{\theta}_D)$ by taking expectations over where $p(\mathbf{X})$ allows us to prioritize where in the input space we want $p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi}) \sim p_{\text{GP}}(\boldsymbol{f}|\boldsymbol{\theta})$ ie $\mathcal{V}_{\mathbf{X}}(\boldsymbol{\phi}_G, \boldsymbol{\theta}_D) = \mathbb{E}_{\mathbf{X} \sim p(\mathbf{X})}\left[\mathcal{V}(\boldsymbol{\phi}_G, \boldsymbol{\theta}_D)\right]$

$$\mathcal{V}_{\mathbf{X}}(\boldsymbol{\phi}_G, \boldsymbol{\theta}_D) \approx \frac{1}{S} \sum_{s=1}^{S} \mathbb{E}_{\mathbf{X} \sim p(\mathbf{X})}\left[\log D(\boldsymbol{f}_{\text{GP}}^{(s)}(\mathbf{X})) + \log\left(1 - D(\boldsymbol{f}_{\text{BNN}}^{(s)}(\mathbf{X}))\right)\right] \tag{8}$$

We will find $(\boldsymbol{\phi}_g^*, \boldsymbol{\theta}_d^*) = \underset{\boldsymbol{\phi}}{\text{argmin}} \ \underset{\boldsymbol{\theta}}{\text{argmax}} \ \mathcal{V}_{\mathbf{X}}(\boldsymbol{\phi}_g, \boldsymbol{\theta}_d)$. This is described in Algorithm 2.

Algorithm 2 Learn the prior through an adversarial algorithm

```
 1: **Initialize** $\boldsymbol{\phi} = \{\boldsymbol{\mu}_{\boldsymbol{\phi}}, \log \boldsymbol{\sigma}_{\boldsymbol{\phi}}\}$ and $\boldsymbol{\theta}_D$
 2: **while** $\boldsymbol{\phi}$ not converged **do**
 3:     $\boldsymbol{\epsilon}^{(s)} \sim p(\boldsymbol{\epsilon}) = \mathcal{N}(\mathbf{0}, \mathbb{I})$    ▷ sample prior noise
 4:     $\mathbf{w}^{(s)} \leftarrow \boldsymbol{t}(\boldsymbol{\phi}, \boldsymbol{\epsilon}^{(s)}) = \boldsymbol{\mu}_{\boldsymbol{\phi}} + \boldsymbol{\sigma}_{\boldsymbol{\phi}} \odot \boldsymbol{\epsilon}^{(s)}$    ▷ sample $S$ weights $\mathbf{w} \sim q(\mathbf{w}|\boldsymbol{\phi})$
 5:     $\mathbf{X} \leftarrow \{\mathbf{x}_1, \dots, \mathbf{x}_n \sim p(\mathbf{x})\}$    ▷ sample data
 6:     $\boldsymbol{f}_{\text{GP}}^{(s)} \leftarrow \mathbf{L}\boldsymbol{\epsilon}^{(s)}$    ▷ sample S function from the GP prior : $\boldsymbol{f}_{\text{GP}}^{(s)} \sim p_{\text{GP}}(\boldsymbol{f})$
 7:     $\boldsymbol{f}_{\text{BNN}}^{(s)} \leftarrow G(\mathbf{X}; \mathbf{w}^{(s)})$    ▷ sample $S$ functions from the BNN prior : $\boldsymbol{f}_{\text{BNN}}^{(s)} \sim p_{\text{BNN}}(\boldsymbol{f})$
 8:     $\mathbf{g}_{\boldsymbol{\phi}} \leftarrow \nabla_{\boldsymbol{\phi}} \mathcal{V}_{\mathbf{X}}(\boldsymbol{\phi}, \boldsymbol{\theta}_d)$    ▷ compute the gradients of the BNN 'generator'
 9:     $\boldsymbol{\phi} \leftarrow \text{adam}(\boldsymbol{\phi}, \mathbf{g})$    ▷ update the BNN 'generator' parameters
10:     $\mathbf{g}_{\boldsymbol{\theta}_d} \leftarrow \nabla_{\boldsymbol{\theta}_d} \mathcal{V}_{\mathbf{X}}(\boldsymbol{\phi}, \boldsymbol{\theta}_d)$    ▷ compute the gradients of the discriminator
11:     $\boldsymbol{\theta}_d \leftarrow \text{adam}(\boldsymbol{\phi}, \mathbf{g}_{\boldsymbol{\theta}_d})$    ▷ update the discriminator parameters
12: **Return** $\boldsymbol{\phi}^*$
```

5 Step Two Maximize the Evidence Lower Bound

Next, we use $\boldsymbol{\phi}^*$ found from step 1 to form a prior on the weights $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}^*_{\boldsymbol{\phi}}, \boldsymbol{\sigma}^*_{\boldsymbol{\phi}}) \equiv q(\mathbf{w}|\boldsymbol{\phi}^*)$. Thereafter we learn the parameters $\boldsymbol{\varphi} = \{\boldsymbol{\mu}_{\boldsymbol{\varphi}}, \log \boldsymbol{\sigma}_{\varphi}\}$ of the variational approximation $q(\mathbf{w}|\boldsymbol{\varphi}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_{\boldsymbol{\varphi}}, \boldsymbol{\sigma}_{\varphi})$ to the true posterior on the weights $p(\mathbf{w}|\mathcal{D})$. To do this we optimize the evidence lower bound (ELBO) $\mathcal{L}(\boldsymbol{\varphi})$ with our optimized prior $p(\mathbf{w}) = q(\mathbf{w}|\boldsymbol{\phi}^*)$

$$\begin{aligned}\mathcal{L}_{\mathcal{D}}(\boldsymbol{\varphi}) &= \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\varphi})}\left[\log p(\mathcal{D}|\mathbf{w})\right] - \mathbb{KL}[q(\mathbf{w}|\boldsymbol{\varphi})||q(\mathbf{w}|\boldsymbol{\phi}^*)] && (9)\\ &= \mathbb{H}[q(\mathbf{w}|\boldsymbol{\varphi})] + \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\varphi})}\left[\log p(\mathcal{D}|\mathbf{w})\right] + \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\varphi})}\left[\log q(\mathbf{w}|\boldsymbol{\phi}^*)\right] && (10)\\ &\approx \log|\boldsymbol{\sigma}_{\varphi}| + \frac{1}{L}\sum_{\ell=1}^{L}\Big[\log p(\mathcal{D}|\mathbf{w}^{(\ell)}) + \log q(\mathbf{w}^{(\ell)}|\boldsymbol{\phi}^*)\Big] && (11)\end{aligned}$$

Where we have used a Monte Carlo estimate of $\mathbb{E}_{q(\mathbf{w}|\boldsymbol{\varphi})}\left[\log p(\mathcal{D}|\mathbf{w}) - \log q(\mathbf{w}^{(\ell)}|\boldsymbol{\phi}^*)\right]$ using L samples $\mathbf{w}^{(\ell)} \sim q(\mathbf{w}|\boldsymbol{\varphi})$. We do this by sampling from a deterministic function of the variational parameters $\boldsymbol{\varphi}$ and noise variables $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) = \mathcal{N}(\mathbf{0}, \mathbb{I})$ such that $\mathbf{w}^{(\ell)} = \boldsymbol{t}(\boldsymbol{\varphi}, \boldsymbol{\epsilon}) = \boldsymbol{\mu}_{\boldsymbol{\varphi}} + \boldsymbol{\sigma}_{\boldsymbol{\varphi}} \odot \boldsymbol{\epsilon}^{(\ell)}$. Thus we can obtain unbiased stochastic gradients of the ELBO with respect to the variational parameters $\boldsymbol{\varphi}$.

$$\mathcal{L}_{\mathcal{D}}(\boldsymbol{\varphi}) = \mathbb{H}[q(\mathbf{w}|\boldsymbol{\varphi})] + \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log p(\mathcal{D}|\boldsymbol{t}(\boldsymbol{\varphi}, \boldsymbol{\epsilon}))\right] - \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log q(\mathbf{w}|\boldsymbol{\phi}^*)\right] \tag{12}$$

We use adam (Kingma and Ba 2015) to optimize (6) and (3).

Algorithm 3 Optimize ELBO

```
1: Require $\boldsymbol{\phi}^*$
2: Initialize $\boldsymbol{\varphi} = \{\boldsymbol{\mu}_{\boldsymbol{\varphi}}, \log \boldsymbol{\sigma}_{\boldsymbol{\varphi}}\}$
3: while $\boldsymbol{\varphi}$ not converged do
4:     $\boldsymbol{\epsilon}^{(\ell)} \sim p(\boldsymbol{\epsilon}) = \mathcal{N}(\mathbf{0}, \mathbb{I})$                ▷ sample noise variables
5:     $\mathbf{w}^{(\ell)} \leftarrow \boldsymbol{g}(\boldsymbol{\varphi}, \boldsymbol{\epsilon}) = \boldsymbol{\mu}_{\boldsymbol{\varphi}} + \boldsymbol{\sigma}_{\boldsymbol{\varphi}} \odot \boldsymbol{\epsilon}^{(\ell)}$    ▷ sample $L$ weights $\mathbf{w} \sim q(\mathbf{w}|\varphi)$
6:     $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\varphi}} \mathcal{L}_{\mathcal{D}}(\boldsymbol{\varphi})$                ▷ compute gradients
7:     $\boldsymbol{\varphi} \leftarrow \text{adam}(\boldsymbol{\varphi}, \mathbf{g})$                ▷ update the parameters
8: Return $\boldsymbol{\varphi}^*$
```

Note that I used variational lower bound or negative free energy in the code not the elbo, its equivalent so i'll leave this be.

6 Computing Gradients in Step One and Two

6.1 Step One by Minimizing the KL

For Gaussian process prior we have $p_{\text{GP}}(\boldsymbol{f}|\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{f}|\mathbf{0}, \mathbf{K})$. Taking the $\log$ gives

$$\log p_{\text{GP}}(\boldsymbol{f}|\boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{f}^\top \mathbf{K}^{-1}\boldsymbol{f} - \frac{1}{2}N \log 2\pi - \frac{1}{2}\log|\mathbf{K}| \propto -\frac{1}{2}\boldsymbol{f}^\top \mathbf{K}^{-1}\boldsymbol{f} \tag{13}$$

by the Cholesky decomposition $\mathbf{K}^{-1} = (\mathbf{L}\mathbf{L}^\top)^{-1} = (\mathbf{L}^\top)^{-1}\mathbf{L}^{-1}$:

$$\mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})}\left[\log p_{\text{GP}}(\boldsymbol{f}^{(s)})\right] \propto -\frac{1}{2}\mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})}\left[(\mathbf{L}^{-1}\boldsymbol{f}^{(s)}(\mathbf{X}))^\top(\mathbf{L}^{-1}\boldsymbol{f}^{(s)}(\mathbf{X}))\right] \tag{14}$$

To compute gradients wrt the prior variational parameters $\boldsymbol{\phi}$ we find

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\mathbf{X}}(\boldsymbol{\phi}) = -\nabla_{\boldsymbol{\phi}}\frac{1}{S}\sum_{s=1}^{S}\mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})}\left[\log p_{\text{GP}}(\boldsymbol{f}^{(s)}(\mathbf{X}))\right],\ \boldsymbol{f}^{(s)} \sim p_{\text{BNN}}(\boldsymbol{f}|\boldsymbol{\phi}) \tag{15}$$

$$= \nabla_{\boldsymbol{\phi}}\frac{1}{S}\sum_{s=1}^{S}\frac{1}{2}\mathbb{E}_{\mathbf{X}\sim p(\mathbf{X})}\left[\boldsymbol{f}^{(s)}(\mathbf{X})^\top\mathbf{K}^{-1}\boldsymbol{f}^{(s)}(\mathbf{X})\right] \tag{16}$$

$$= \frac{1}{S}\sum_{s=1}^{S}\frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\nabla_{\boldsymbol{\phi}}\left[\mathbf{L}^{-1}\boldsymbol{f}(\mathbf{X}_n, \mathbf{w}^{(s)})\right]^\top\left[\mathbf{L}^{-1}\boldsymbol{f}(\mathbf{X}_n, \mathbf{w}^{(s)})\right] \tag{17}$$

$$\tag{18}$$

Where we have made use of the reparametrization trick $\mathbf{w}^{(s)} = \boldsymbol{g}(\boldsymbol{\phi}, \boldsymbol{\epsilon}^{(s)})$

6.2 Step Two