Meta Learning Better Bayesian Priors

Victor Armegioiu

Introduction

Bayesian inference has been a long standing focus in Machine-Learning/Mathematical Statistics as a fundamental framework for more precise inference via incorporating prior knowledge about the parameter distribution, and for providing *epistemic uncertainty* estimates (uncertainty due to the model parameters). In the case of *Bayesian Neural Networks* (BNNs) as shown in Blundell et al. [2015], both priors and approximate posteriors on neural net parameters are specified as mixtures of isotropic Gaussian distributions.

While computationally convenient, the regularizing effect stemming from Gaussian priors is limited. Firstly, there is no theoretical or empirical evidence showing that Gaussian priors are a generally sound choice, regardless of the underlying problem. Naturally, forcing the parameter distribution to be Gaussian will be bound to limit the model performance due to the lack of expressiveness in the prior. Secondly, searching for appropriate families of distributions in the parameter space of neural nets is extremely challenging due to obvious computational limitations.

In order to challenge these shortcomings, we will investigate whether we can inject more useful inductive biases into the training procedure by meta-learning a prior from several sets of related tasks, in a similar fashion to Rothfuss et al. [2020]; Amit and Meir [2018]. However, the critical difference is that we aim our efforts at showing that it is possible to learn useful priors in the *function space*, instead of the weight space. This alleviates the over-parameterization issues, and allows for specifying arbitrarily rich priors as sets of particles.

Stein Variational Gradient Descent

Stein Variational Gradient Descent (SVGD) [Liu and Wang, 2016] is a relatively recent sampling technique based on iteratively transporting particles to match a target distribution. As a brief overview, let ν be a target probability measure with density $p(x)$ supported on a subset $X \subset \mathbb{R}^d$.
We seek to approximate the target measure via a finite set of particles $\{x_i\}_{i=1}^n$, inducing the empirical measure $\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$. This is done via incrementally transporting particles using the following update rule:

$$T(x) = x + \epsilon \cdot \phi(x)$$

Here, the vector field $\phi(\cdot)$ denotes the optimal perturbation direction, moving particles towards the target measure, and ϵ is a small, adjustable step size. It can be shown that the optimal $\phi(\cdot)$ can be obtained as a solution to the problem

$$\max_{\phi \in \mathcal{H}} \left\{ -\frac{d}{d\epsilon}\mathrm{KL}(T\mu \| \nu)|_{\epsilon=0}, \;\; s.t. \;\; \|\phi\|_{\mathcal{H} \leq 1} \right\}$$

The functional objective reaches its optimum at a function ϕ in some 1-ball induced by the norm topology of $\mathcal{H}$, such that the KL divergence between the pushforward measure $T\mu$

and the target measure ν is minimized. Liu and Wang [2016] show that the optimal velocity field ϕ can be written as:

$$\phi^*_{\mu,p}(\cdot) := \mathbb{E}_{x\sim\mu}[\nabla \log p(x) k(x,\cdot) + \nabla_x k(x,\cdot)] \tag{1}$$

Here, $k(\cdot,\cdot)$ is a positive semi-definite kernel which induces the *Reproducing Kernel Hilbert Space* (RKHS) $\mathcal{H}$. On an intuitive level, the score term inside the expectation pushes particles towards high density regions of the target measure, while the second term acts as a repulsive term, forcing particles to move away from each other, hence covering multiple modes. In practice, the expectation is restricted to a sum over the particles. This gives rise to the following update rule:

$$x_i \leftarrow x_i + \epsilon \cdot \hat{\phi}(x_i) \quad \text{such that} \quad \hat{\phi}(x) = \frac{1}{n}\sum_{j=1}^{n} k(x, x_j)\nabla_{x_j} \log p(x_j) + \nabla_{x_j} k(x_j, x) \tag{2}$$

Perhaps unsurprisingly, SVGD offers a principled way of representing the posterior over neural net parameters $p(\theta|\mathcal{D})$ as an empirical distribution, or a set of particles. Indeed one can choose randomly initialized base learners $\theta_1, \ldots, \theta_k$ and drive them towards high density areas of $\log p(\mathcal{D}|\{\theta_i\}_{i=1}^k)$ via SVGD, while also ensuring they do not simply collapse to the MAP estimate, and instead converge to different modes (as constrained to do via the repulsive force terms).

Naturally, this leads to better generalization than typical BNNs with Gaussian mixture priors over their weights - however, the number of particles k is a critical hyperparameter which needs to be tuned, since simply increasing the number of particles will not improve performance as the true posterior over weights only has a finite number of modes.

Gradient Estimation for Implicit Distributions

This section is concerned with examining candidate methods for evaluating score functions $\nabla_f \log p(f)$, where p is specified as an empirical measure $p := \sum_i \delta_{f_i}$. This will be of utmost importance in computing gradients of the functional evidence lower bound objective function when the prior is specified as a set of particles. We will present two existing methods, one parametric (based on training a neural net to do score estimations) and one non-parametric (leverages eigenfunction approximations in Reproducing Kernel Hilbert spaces induced by some kernel function $k(\cdot,\cdot)$).

Spectral Stein Gradient Estimator

The *Spectral Stein Gradient Estimator* (SSGE) Shi et al. [2018] is a tractable method for gradient estimation of score functions $\nabla_x \log q(x)$, which only requires access to samples $x \sim q$, hence an expression for the density is not needed.

This is done by leveraging the fact that for any given positive definite kernel $k(\cdot,\cdot)$ in the Stein class of q, one has the following eigenfunction expansion $k(x,y) = \sum_i \mu_i \psi_i(x)\psi_i(y)$ via Mercer's theorem. We note that the eigenfunctions $\{\psi_i\}_{i\geq 1}, \psi_i \in L^2(\mathcal{X}, q)$ are orthogonal with respect to the underlying density q, in that $\int \psi_i(x)\psi_j(x)q(x)dx = \delta_i^j$. Note that $L^2(\mathcal{X}, q)$ simply denotes the space of all square integrable functions over $\mathcal{X}$ w.r.t. the density $q(\cdot)$.

Shi et al. [2018] show how to obtain a gradient estimator via an eigenfunction decomposition of the partial derivatives $\{g_i := \frac{\partial}{\partial x_i}\}_{i=1}^d$ where $g := \nabla_x \log p(\cdot)$ and $d := \dim(\mathcal{X})$. Assuming $g_1, \ldots, g_d \in L_2(\mathcal{X}, q)$ we can expand them into a spectral series:

$$g_i(x) = \sum_{i=1}^{\infty} \beta_{ij} \psi_j(x) \tag{3}$$

It can be shown that the coefficients β_{ij} can be recovered as

$$\beta_{ij} = -\sum_{j \geq 1} \mathbb{E}_q[\nabla_{x_i} \psi_j(x)] \tag{4}$$

The eigenfunctions and their respective derivatives are approximated via the Nystrom [1930] method. The equation above is made tractable by restricting the sum to a finite number of terms and using Monte Carlo estimates instead of the full expectation over q.

Sliced Score Matching

Sliced score matching (SSM) is a parametric method proposed in Song et al. [2020] for approximating score functions of implicit distributions. Assuming access to samples $x_1, \ldots, x_n \in \mathbb{R}^d$, from some distribution $p_d(\cdot)$, we seek to learn an unnormalized distribution $\tilde{p}_m(x;\theta)$ where θ is the set of parameters of our density estimation model. The normalized density under our model can be written as

$$p_m(x;\theta) := \frac{\tilde{p}_m(x;\theta)}{\int_{\mathcal{X}} \tilde{p}_m(x;\theta)dx} \tag{5}$$

Fortunately, $\nabla_x \log p_m(x;\theta) = \nabla_x \tilde{p}_m(x;\theta) - \nabla_x \int_{\mathcal{X}} \log \tilde{p}_m(x;\theta)dx = \nabla_x \log \tilde{p}_m(x;\theta)$. Denoting $s_m(x;\theta) := \nabla_x \log p_m(x;\theta)$ and $s_d(x) := \nabla_x \log p_d(x)$ the authors show that the objective

$$L(\theta) := \frac{1}{2} \mathbb{E}_{x \sim p_d}[\|s_m(x;\theta) - s_d(x)\|_2^2] \tag{6}$$

can be rewritten via integration by parts in order to get rid of the unavailable/intractable score function $s_d(\cdot)$ as

$$J(\theta) := \mathbb{E}_{x \sim p_d}\left[\operatorname{tr}(\nabla_x s_m(x;\theta)) + \frac{1}{2}\|s_m(x;\theta)\|_2^2\right] \tag{7}$$

where $\operatorname{tr}(\cdot)$ denotes the trace operator and $\nabla_x p(x;\theta)$ is the Hessian matrix of $\log \tilde{p}_m(x;\theta)$. Unfortunately, the trace of the Hessian term cannot be easily computed - it would require applying backpropagation d times to $s_m(x;\theta)$ where d can get arbitrarily large. In order to circumvent this issue, the authors propose the reformulation

$$L(\theta; p_v) := \mathbb{E}_{v \sim p_v} \mathbb{E}_{x \sim p_d}[\|v^T s_m(x;\theta) - v^T s_d(x)\|_2^2] \tag{8}$$

where p_v is either a Rademacher or a multivariate Gaussian distribution. This new objective ensures that the score s_m and s_d are close in Euclidean distance by making their projections on random vectors as close as possible. In order to eliminate the intractable $s_d(\cdot)$ term the authors apply the same integration by parts trick used to derive eq. (7) from eq. (6), arriving at the feasible objective

$$\mathbb{E}_{v \sim p_v} \mathbb{E}_{x \sim p_d}[v^T \nabla_x s_m(x;\theta) v + \frac{1}{2}(v^T s_m(x;\theta))^2] \tag{9}$$

which is easily made computationally tractable via drawing Monte Carlo estimates instead of computing the full expectations. Note that one can replace $s_m(x;\theta)$ by a neural network $h(x;\theta)$ and obtain therefore a feasible score function learning algorithm. However, one should note that $h(x;\theta)$ might not correspond to a gradient vector (hence it might not be orthogonal to level sets) - however it is still useful for our purposes since the Euclidean distance from the true gradient will be indeed minimized.

Informative Priors Represented as Particles

The role of priors is to inject useful inductive biases into the model structure. However, as is the case for Bayesian Neural Networks, the prior is typically chosen as an isotropic Gaussian, which encodes little useful knowledge about the problem at hand, while also imposing a hard restriction over the shape of the true prior, which is generally much more complex than a simple Gaussian.

A more informed route is to learn what priors to use directly from the data. This would correspond to a form of meta-regularization, such that inductive biases incorporated in the priors would be learned directly from the true generating process underlying the meta-training dataset. Hence, we will experiment with representing our prior as a set of particles $\{\theta_i\}_{i=1}^n$.

Functional Variational Inference

Consider the typical regression problem, where we model the likelihood of a given label as $p(y|f(x)) = \mathcal{N}(y|f(x), \sigma_y^2)$. Adopting the function space view, let f^X, denote the function values corresponding to the inputs $X \in \mathcal{X}^n$, where $\mathcal{X}^n$ is some arbitrary measurable index set.
Sun et al. [2019] show how to adapt the typical evidence lower bound - as commonly used in weight space training for Bayesian Neural Networks - to the function space. Unfortunately, these is no Lebesgue measure that can be constructed for infinitely dimensional Hilbert spaces (apart from the zero measure and counting measures, which are not very useful in our case), hence the functional KL-divergence is not straight forward to define as $\mathrm{KL}(q\|p) := \int q(f) \log \frac{q(f)}{p(f)} df$. They show instead that for stochastic processes P, Q, indexed by $\mathcal{X}^n$ the following holds

$$\mathrm{KL}(Q\|P) := \sup_{n\in \mathrm{N}, X\in\mathcal{X}^n} \mathrm{KL}(Q_X\|P_X) \tag{10}$$

By letting the variational posterior q_ϕ be parameterized by a stochastic neural net, sampling functions from q_ϕ is then done by evaluating the NN at a given location x, i.e. $f(x) = g_\phi(x;\xi)$ where g_ϕ is the neural approximator, and ξ is a random vector used for the reparameterization trick. The functional prior is denoted by $p(\cdot)$. Using the reformulation of ELBO in function space, the new objective, fELBO, takes the familiar form

$$\mathcal{L}(q) = \mathrm{E}_q[\log p(\mathcal{D}|f)] - \sup_{n\in \mathrm{N}, X\in\mathcal{X}^n} \mathrm{KL}(q(f^X)\|p(f^X)) \tag{11}$$

Computational Considerations in Minimizing fELBO

As there is an uncountably infinite number of index subsets to consider, evaluating the KL divergence for stochastic processes with limited data and time requires a different optimization scheme. This is made tractable by noting that the supremum in eq. (10) can be rewritten in terms of an expectation over index sets instead.

Assuming $\mathcal{X}^n$ is a compact space, the *Heine-Borel theorem* asserts that there is a finite subcover S for any open cover C of $\mathcal{X}^n$, such that $\mathcal{X}^n = \cup_{X\in S} X$. Given access to a sampling distribution $c(\cdot)$ over *finite measurement sets*, such that $X \sim c(\cdot)$,$\forall X \in S$ the KL term may be rewritten as

$$\sup_{n\in \mathrm{N}, X\in\mathcal{X}^n} \mathrm{KL}(q(f^X)\|p(f^X)) = \mathbb{E}_{X\sim c(\cdot)}[\mathrm{KL}(q(f^X)\|p(f^X))] \tag{12}$$

Other computational difficulties arise due to the fact that optimizing the objective function $\mathcal{L}(q)$, requires computing the gradients of the KL term with respect to the function values f

and the parameters ϕ (via the chain rule). As shown in Sun et al. [2019], the KL gradients may be tractably expressed as an expectation

$$\nabla_\phi \mathrm{KL}(q \| p) = \mathbb{E}_{\xi \sim \mathcal{N}(0,I)}[\nabla_\phi f(\nabla_f \log q(f) - \nabla_f \log p(f))]. \tag{13}$$

Score functions for the prior may be exactly computed when assuming analytic pdfs such as GP priors. However, this is particularly problematic when we represent the prior as a set of particles in function space, as we will not have direct access to its score function $\nabla_f \log p(f^X)$, which is needed for minimizing the KL divergence. Furthermore, since drawing functions from the posterior is done by drawing different neural networks from the variational posterior, and evaluating them at a given input location, we also need to use a score estimator for the resulting posterior empirical measure as well (since we do not have any density estimation mechanism for this).

In these non-analytical cases, where we are only given access to implicit distributions, we rely on score estimators such as SSGE and SSM . In terms of training procedures we use algorithm 2, algorithm 3 which we will introduce shortly after explaining the meta-training environment.

Meta Learning Functions

We assume a typical meta-learning scenario, where one has access to a limited number of related tasks which are assumed to be generated under the same generating process. Naturally, we expect the underlying generating parameters of each task to be subject to small perturbations, rendering a family of functions. Specifically, we assume access to a *meta-training* dataset $\mathcal{D}_{\text{train}} := \{T_i\}_{i=1}^m$ where each task $T_i := \{(x_j, y_j)\}_{j=1}^k$ corresponds to a meta-learning task. Analogously, we assume access to a *meta-tuning* dataset $\mathcal{D}_{\text{tune}} := \{T_i\}_{i=1}^n$, where typically $n \ll m$.

This corresponds to real-life scenarios where one has to learn a model on a limited set of training tasks (the meta-tuning dataset), while having access to a significantly larger set of related tasks (the meta-training dataset). One specific example would be a robotics task, where one seeks to use reinforcement learning to train a robotic hand to perform various picking-up/moving around objects tasks. In this context one might have access to an unlimited (or very high) amount of meta-training tasks extracted from a simulator, while only having access to very few real life meta-tuning examples.

Meta-Learning Functional Priors

Assuming the previous meta-training/tuning setting, we differentiate between analytical priors and non-analytical priors represented as sets of particles.

In the analytic context, we assume function values can be captured reasonably well by a *Gaussian Process*, $f \sim \mathcal{GP}(m_\theta(\cdot), k_\phi(\cdot, \cdot))$. Apart from the kernel parameters ϕ, we find that useful inductive biases can also be baked into a learnable mean function m_θ (it is trivial to construct examples where the mean function is non-zero, and having this knowledge a priori would naturally lead to better generalization). We shall only use the meta-training tasks $\mathcal{D}_{\text{train}}$ for fitting the GP parameters, hence we run no risk of overfitting on the actual tasks we are interested in learning in (the meta-tuning tasks $\mathcal{D}_{\text{tune}}$). Algorithm 1 shows the training procedure for meta-learning the GP prior parameters

Algorithm 1 GP Prior Meta Learning

```
Require: mean function neural net $m_\theta(\cdot)$, PSD kernel $k_\phi(\cdot,\cdot)$.
Require: meta-training dataset of tasks $\mathcal{D}_{\text{train}}$.
  while not converged do
    for $(x_i, y_i) \in \mathcal{D}_{\text{train}}$ do
      $\theta, \phi \leftarrow$ OPTIMIZER($\theta, \phi; \nabla_{\phi,\theta} \log p(y_i|x_i;\theta,\phi)$)    // $\log p(y_i|x_i;\theta,\phi)$ is the NLL.
    end for
  end while
```

In the non-analytic context, we impose no parametric restrictions on the shape of the prior distribution, and will use the particles themselves instead. Let $\mathcal{Y} := \text{vec}(\{y_i\} \in \mathcal{D}_{\text{train}})$, that is, a column vector containing all function values corresponding to the meta-training tasks. We then represent the prior as the empirical measure $p_T := \sum_{t=1}^{T:=|\mathcal{Y}|} \delta_{y_t}$ - note that $p_T \underset{T\to\infty}{\to} p$, i.e. p_T converges weakly to the true prior p when $T \to \infty$.

The trade-offs are evident in both cases - the GP approach is limited in expressiveness by its kernel function, which will not capture the true nature of the function values unless there is a clear manifestation of periodicity. Furthermore, the Gaussian assumption is generally not supported by real world data. However, this approach lends itself to easy and exact log probability gradients via backpropagation.

In the case of the empirical measure approach, using particles allows us to make full use of the meta-training data and maximize the expressiveness of our representation of the prior. However, this is only the case when we have a large enough number of samples in order to assume closeness to the true prior. Another issue is that we have to introduce an extra layer of approximate computations for computing log-prior gradients (SSGE, SSM). Both these estimators introduce high variance and loss of precision when sufficient data is not available.

Deploying the Meta-Learned Functional Prior

In order to incorporate useful inductive biases in our model training, we propose adapting the fELBO objective formulation to use a functional prior $p(\cdot)$ which is extrapolated from the meta-training data $\mathcal{D}_{\text{train}}$. The functional posterior $q(\cdot)$ can be specified by a typical BNN with Gaussian priors over network parameters (as done in Sun et al. [2019]), or as a set of particles approximating the true posterior over weights. In the second approach the particles (base learners) are trained with SVGD.

Algorithm 2 fELBO Optimization - variational posterior is a weight space BNN that is trained via standard variational inference.

```
Require: variational posterior over functions $q_\phi(\cdot)$, prior over functions $p(\cdot)$.
Require: meta-tuning tasks $\mathcal{D}_{\text{tune}}$.
Require: KL term weight $\lambda$.
  while not converged do
    for $(x_i, y_i) \in \mathcal{D}_{\text{tune}}$ do
      draw $k$ function values $f_t^{x_i} \sim q_\phi(x_i, \xi_t), t \in [k]$
      let $g_{LL} := \nabla_\phi \sum_{t,(x,y)\in(x_i,y_i)} \log p(y|f_t(x))$
      let $g_{KL} := \nabla_\phi \text{KL}(q\|p) = \mathbb{E}_{\xi\sim\mathcal{N}(0,I)}[\nabla_\phi f(\nabla_f \log q(f) - \nabla_f \log p(f))]$
      $\phi \leftarrow$ OPTIMIZER($\phi; g_{LL} - \lambda g_{KL}$)
    end for
  end while
```

Algorithm 3 fELBO Optimization - variational posterior is a set of particles (neural nets) trained with SVGD.

```
Require: particles $\theta_1, \dots, \theta_k$ approximating posterior over weights , prior over functions $p(\cdot)$.
Require: PSD kernel $k(\cdot, \cdot)$.
Require: meta-tuning tasks $\mathcal{D}_{\text{tune}}$, KL term weight $\lambda$, step size $\epsilon$.
  while not converged do
    for $(x_i, y_i) \in \mathcal{D}_{\text{tune}}$ do
      evaluate particles $f_t^{x_i} := \{\theta_t(x_i)\}_{t=1}^k$
      for $t \in [k]$ do
        let $g_{SVGD}[t] := \frac{\epsilon}{k} \sum_{j=1}^k k(\theta_t, \theta_j) \nabla_{\theta_j} \log p(y_i|\theta_j(x_i)) + \nabla_{\theta_j} k(\theta_j, \theta_t)$
        let $g_{KL}[t] := \nabla_{\theta_t} \text{KL}(q\|p) = \mathbb{E}_{\xi \sim \mathcal{N}(0,I)}[\nabla_{\theta_t} f(\nabla_f \log q(f_t^{x_i}) - \nabla_f \log p(f_t^{x_i}))]$
        $\theta_t \leftarrow \text{OPTIMIZER}(\theta_t; g_{SVGD}[t] - \lambda g_{KL}[t])$
      end for
    end for
  end while
```

Experimental Setup

In order to be able to perform meta-training/tuning we used the *sinusoid* environment for generating related tasks. We approximate a finite cover of the compact index set $\mathcal{X} := [-5, 5]$ via Monte Carlo estimates, i.e. we define the sampling distribution over finite measurement sets as $c(\cdot) := \mathcal{U}(-5, 5)$ and draw the index sets as $X \sim c(\cdot)$ for any given meta-task. The function values at the given input locations

$$y := f_{\beta,a,b,c,d}(x) = \beta \cdot x + a \cdot \sin(1.5 \cdot (x - b)) + c \tag{14}$$

are generated via summing an affine transform of the inputs with a scaled/shifted sinusoid. Task parameters are different for each particular task, and are generated as

$$\beta \sim \mathcal{N}(0.5, 0.2^2),\ a \sim \mathcal{U}(0.7, 1.3),\ b \sim \mathcal{N}(0, 0.1^2),\ c \sim \mathcal{N}(5.0, 0.1^2) \tag{15}$$

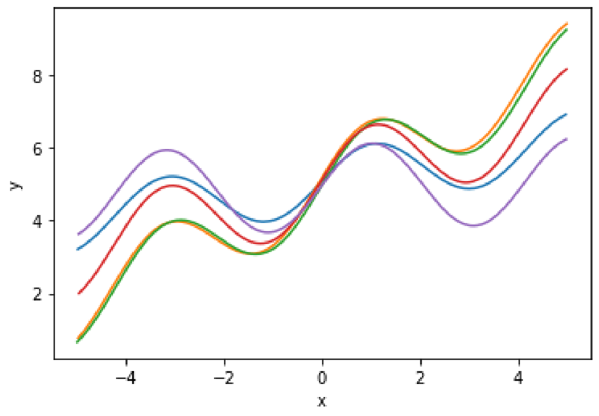

Figure 1: 5 functions drawn from the sinusoid environment.

Figure 1 shows a sample realization of 5 tasks drawn from the sinusoid environment. As per eq. (14), eq. (15), these sample functions will be used as a particle approximation to $p(f)$ (our prior over functions).

For our training purposes we use $[50, 100]$ tasks for the meta-training dataset $\mathcal{D}_{\text{train}}$ and $[25, 30]$ tasks for the meta-tuning dataset $\mathcal{D}_{\text{tune}}$, out of which we held out 5 for testing. We draw 5 samples per task. The training consists of 2 phases: constructing the functional prior distribution using the meta-training tasks $\mathcal{D}_{\text{train}}$, and deploying the prior knowledge in the fELBO training procedure as outlined in algorithm 2, algorithm 3.

Results

In order to assess the usefulness of the functional priors, we compare the log-likelihood of the data under models trained to just maximize MLE (i.e. applying algorithm 2, algorithm 3 and setting the KL weight $\lambda := 0$), versus models trained to optimize the full fELBO objective ($\lambda := 1$). We find that regardless of the underlying BNN posterior representation (trained via standard VI or SVGD), using functional priors not only always improved likelihood results, but the training converged much faster as well. Unfortunately, there is high variance in the performances of the different gradient estimators depending on the underlying training method used to represent the variational posterior and the size of the available meta-training dataset.

Variational Posterior Training	Prior Score Function Estimation	Meta-Test Tasks Log-Likelihood (T 95% confidence interval)
BNN + Standard VI	GP Prior	$(-0.35, 0.01)$
BNN + Standard VI	SSGE	**(-0.06, -0.05)**
BNN + Standard VI	SSM	$(-0.22, 0.019)$
BNN + Standard VI	No functional prior	$(-0.25, 0.05)$

Variational Posterior Training	Prior Score Function Estimation	Meta-Test Tasks Log-Likelihood (T 95% confidence interval)
Particles + SVGD	GP Prior	$(-0.14, 0.02)$
Particles + SVGD	SSGE	$(-0.19, -0.14)$
Particles + SVGD	SSM	**(-0.03, -0.01)**
Particles + SVGD	No functional prior	$(-0.19, -0.09)$

The 'Variational Posterior Training' column shows which method is used for defining the variational posterior (as per Algorithms 3, 2). In the 'BNN + Standard VI' case, we simply assume a weight space fully factorized Gaussian posterior approximation $q(\phi) = \mathcal{N}(\phi, \sigma^2 I)$, with a Gaussian prior over weights $p(w) = \mathcal{N}(0, I)$. The optimal location parameters ϕ are found by solving

$$\phi^* := \arg\min_{\phi} \; \mathrm{KL}(q(w|\phi) \;||\; p(w)) - \mathbb{E}_{q(w|\phi)}[\log p(\mathcal{D}|w)] \tag{16}$$

The 'Prior Score Function Estimation' specifies what method was used for computing/approximating $\nabla_f \log p(f)$ (just backpropagation for GP Priors and SSGE, SSM for implicit priors). Note that not using functional priors in the 'BNN + Standard VI' case simply amounts to training a BNN via Bayes By Backprop (BBB) (Blundell et al. [2015]). The last column shows log-likelihood estimates under the several different settings (higher is better) and corresponding .95 confidence intervals.

We see that just maximizing MLE and discounting functional prior information (the 'No functional prior' entries) always leads to significantly worse results, showing empirically as well that functional priors do indeed encode extremely useful inductive biases that can lead to better generalization.

References

R. Amit and R. Meir. Meta-learning by adjusting priors based on extended pac-bayes theory. In *International Conference on Machine Learning*, pages 205–214. PMLR, 2018.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR, 2015.

Q. Liu and D. Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in neural information processing systems*, pages 2378–2386, 2016.

Nystrom. "U about the practical solution ö solution of integral equations with applications to boundary value problems. *Acta Mathematica*, 54(1):185–204, 1930.

J. Rothfuss, V. Fortuin, M. Josifoski, and A. Krause. Pacoh: Bayes-optimal meta-learning with pac-guarantees. *arXiv preprint arXiv:2002.05551*, 2020.

J. Shi, S. Sun, and J. Zhu. A spectral approach to gradient estimation for implicit distributions. *arXiv preprint arXiv:1806.02925*, 2018.

Y. Song, S. Garg, J. Shi, and S. Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR, 2020.

S. Sun, G. Zhang, J. Shi, and R. Grosse. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.