ALPaCA vs. GP-based Prior Learning: A Comparison between two Bayesian Meta-Learning Algorithms

Yilun Wu
D-MAVT
ETH Zürich
Switzerland, ZH 8001
wuyil@student.ethz.ch

Abstract

Meta-learning or few-shot learning, has been successfully applied in a wide range of domains from computer vision to reinforcement learning. Among the many frameworks proposed for meta-learning, bayesian methods are particularly favoured when accurate and calibrated uncertainty estimate is required. In this paper, we investigate the similarities and disparities among two recently published bayesian meta-learning methods: ALPaCA (Harrison et al. [2018]) and PACOH (Rothfuss et al. [2020]). We provide theoretical analysis as well as empirical benchmarks across synthetic and real-world dataset. While ALPaCA holds advantage in computation time by the usage of a linear kernel, general GP-based methods provide much more flexibility and achieves better result across datasets when using a common kernel such as SE (Squared Exponential) kernel. The influence of different loss function choice is also discussed.

1 Introduction

Many sources such as Mnih et al. [2015] and Dubey et al. [2018] suggest what make humans so good at learning to solve new tasks is by efficiently extracting prior knowledge from past experience and leveraging them during decision making.

Acquiring such generalizable inductive bias has always been a central problem in machine learning (Baxter [2000]), especially under meta-learning settings where the amount of online learning data is limited (Caruana [1997], Vanschoren [2018]). However, adapting a highly complex model based on a few training input is inherently challenging and produces a large uncertainty if not well regulated (e.g. meta-overfitting reported by Mishra et al. [2017] and Rothfuss et al. [2020]). To improve the robustness of meta-learning methods, it is therefore a good idea to incorporate uncertainty statistics as a metric during training and evaluation in the learning process.

Bayesian methods are very popular in this regime, since it provides meaningful and accurate statistics through well-understood update rule. Both parametric methods such as Bayesian Neural Network and Non-parametric methods such as GP (Gaussian Processes) have seen tremendous successful real-world applications such as Berkenkamp et al. [2017], Hewing et al. [2019] and Peretroukhin et al. [2017]. The representation power of standard GP methods are furthermore extended by embedding function approximators such as deep neural networks in its kernel (Wilson et al. [2015]) and mean function (Fortuin et al. [2019]) or jointly (Calandra et al. [2016] and Garnelo et al. [2018]). It is therefore natural to adopt bayesian statistics in the scope of meta-learning where bayesian priors are used as the inductive bias at the task-level. Recent adoptions in meta-learning literatures include Kim et al. [2018], Grant et al. [2018], Ravi and Beatson [2019].

In this work, we focus on comparing two similar bayesian learning frameworks: BLR (Bayesian Linear Regression) vs. GPR (Gaussian Process Regression) as published in Harrison et al. [2018] and Rothfuss et al. [2020]. We show an one-to-one correspondence between BLR models and GPR models in section 2.3, in the sense that BLR and GPR models share the same hypothesis space. We later show the objective function used in Harrison et al. [2018], as a result of directly minimizing the conditional (posterior) KL divergence between the true distribution and predicted distribution, under mild assumptions, is the same as maximizing the likelihood of overall dataset. In section 3, we present empirical results on models consisting of different architectures and loss function with varying nuances. A trade-off between performance and computation still exists between kernel-based GP methods and bayesian linear regression, despite opposite claims from Harrison et al. [2018].

2 Theoratical Equivalence and Disparity between ALPaCA and GPR Meta Learning

In this section, we point out the theoretical equivalence and differences between ALPaCA Harrison et al. [2018] and PACOH Rothfuss et al. [2020] in the meta learning setting.

We first show that during inference (online update), the posterior predictive distribution of response variable conditioned on context data from BLR (Bayesian Linear Regression) in latent space is the same as the posterior predictive distribution derived from a GPR (Gaussian Process Regression) with a special linear kernel.

We then compare the loss function used for learning the parameters constituting the meta prior in two methods. It can be shown that under the assumption of uniform sampling of context time horizon, the two loss functions are equivalent to each other.

2.1 Latent Space Bayesian Linear Regression

We summarize the meta-learning architecture used in ALPaCA as bayesian linear regression with transformed feature space via basis function $\phi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_\phi}$. This is usually implemented with a deep neural network with n_ϕ hidden layers in the last layer, as in Harrison et al. [2018]. The dependent variable y is linearly regressed as: $y^\top = \phi(x)^\top \mathbf{K} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \mathbf{\Sigma}_\epsilon), \mathbf{K} \in \mathbb{R}^{n_\phi \times n_y}$. Note, in this case, both $x, \phi(x)$ and y are multi-dimensional vectors with dimension n_x, n_ϕ, n_y respectively.

Assume a prior for $\mathbf{K} \sim \mathcal{MN}(\mathbf{K_0}, \Lambda_0^{-1}, \Sigma_\epsilon)$ where $\mathcal{MN}$ indicates matrix normal distribution.[1] With the likelihood (conditional probability) of data Y being $p(Y|\Phi, \mathbf{K}, \Sigma_\epsilon) \propto |\Sigma_\epsilon|^{-n_y/2} \exp(-\frac{1}{2}\operatorname{tr}((Y - \Phi\mathbf{K})\Sigma_\epsilon^{-1}(Y - \Phi\mathbf{K})^\top))$, where $\Phi = [\phi(x_1), \phi(x_2), ...]$, we arrive at the posterior distribution of $\mathbf{K}$: $p(\mathbf{K}|Y, X) = \mathcal{MN}(\mathbf{K}_\tau, \Lambda_\tau^{-1}, \Sigma_\epsilon)$, which is of the same distribution family as the prior, making matrix normal distribution the conjugate prior for the likelihood function above.

Plugging the posterior distribution of $\mathbf{K}$ into the likelihood function, we obtain posterior predictive distribution of Y_t at test points X_t conditioned on observed context data (X_c, Y_c):

$$p(Y_t|X_t, X_c, Y_c) = \mathcal{N}(\Phi(X_t)^\top \mathbf{K}_\tau, \Sigma_\tau) \tag{1}$$

where

$$\mathbf{K}_\tau = \Lambda_\tau^{-1}(\Phi(X_c)Y_c + \Lambda_0 \mathbf{K_0}) \tag{2}$$

$$\Lambda_\tau = \Phi(X_c)\Phi(X_c)^\top + \Lambda_0 \tag{3}$$

$$\Sigma_\tau = (\mathbf{I} + \Phi(X_t)^\top \Lambda_\tau^{-1}\Phi(X_t)) \otimes \Sigma_\epsilon \tag{4}$$

where $\otimes$ denotes the Kronecker product.[2]

[1]The definition of the matrix normal distribution can be found at `https://en.wikipedia.org/wiki/Matrix_normal_distribution`.

[2]The definition of Kronecker product can be found at `https://en.wikipedia.org/wiki/Kronecker_product`.

2.2 Gaussian Process Regression with Linear Kernel

Now consider multi-output (multi-task) Gaussian Process Regression $f: X \in \mathbb{R}^{n_x} \to Y \in \mathbb{R}^{n_y} \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot))$ where $m(\cdot): X \in \mathbb{R}^{n_x} \to Y \in \mathbb{R}^{n_y}$ is the mean function and $k(\cdot,\cdot): X \times X \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \Sigma \in \mathbb{R}^{n_y \times n_y}$ is the kernel function. Note in this case, the output of f is multi-dimensional, thus the output of the kernel function is a matrix instead of a scalar value. The matrix output of the kernel function can be interpreted as the cross-covariance matrix at two different intputs under their prior distribution: $k(x, x') = \mathbb{E}[(f(x) - \overline{f(x)})(f(x') - \overline{f(x')})^\top]$.

Let $m(x) = \mathbf{K_0}^\top \phi(x)$ and $k(x, x') = \phi(x)^\top \Lambda_0^{-1} \phi(x') \Sigma_\epsilon$ with the same $\mathbf{K_0}, \Lambda_0, \Sigma_\epsilon$ used in defining the prior distribution of $\mathbf{K}$ in 2.1. One can construct the gram matrix of a set of points $X = [x_1, x_2, ..., x_n]^\top$ by stacking pairs of kernel functions:

$$G(X) = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \dots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \dots & k(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \dots & k(x_n,x_n) \end{bmatrix} \tag{5}$$

$$= \Phi(X)^\top \Lambda_0^{-1} \Phi(X) \otimes \Sigma_\epsilon \tag{6}$$

Further let $y(x) = f(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \Sigma_\epsilon)$. Then the prior distribution for $Y(X) = [y(x_1), y(x_2), ..., y(x_n)]^\top$ reads as follows:

$$Y \sim \mathcal{N}(m(X), (G(X) + \mathbf{I}) \otimes \Sigma_\epsilon) \tag{7}$$

From this we can write the following prior distribution for target values at context points X_c and test points X_t:

$$\begin{bmatrix} Y(X_c) \\ Y(X_t) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right) \tag{8}$$

where

$$\mu_1 = \Phi(X_c)^\top \mathbf{K_0},\ \mu_2 = \Phi(X_t)^\top \mathbf{K_0} \tag{9}$$

$$\Sigma_{11} = (\Phi(X_c)^\top \Lambda_0^{-1} \Phi(X_c) + \mathbf{I}) \otimes \Sigma_\epsilon,\ \Sigma_{22} = (\Phi(X_t)^\top \Lambda_0^{-1} \Phi(X_t) + \mathbf{I}) \otimes \Sigma_\epsilon \tag{10}$$

$$\Sigma_{12} = \Sigma_{21}^\top = (\Phi(X_c)^\top \Lambda_0^{-1} \Phi(X_t)) \otimes \Sigma_\epsilon \tag{11}$$

By applying marginalization rule, we arrive at the following posterior distribution for Y_t at test points X_t:

$$p(Y_t | X_t, X_c, Y_c = z) = \mathcal{N}(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(z - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}) \tag{12}$$

$$\begin{aligned} =&\mathcal{N}\{\Phi(X_t)^\top \mathbf{K_0} + \Phi(X_t)^\top \Lambda_0^{-1} \Phi(X_c)(\Phi(X_c)^\top \Lambda_0^{-1} \Phi(X_c) + \mathbf{I})^{-1}(Y_c - \Phi(X_c)^\top \mathbf{K_0}), \\ &[\Phi(X_t)^\top \Lambda_0^{-1} \Phi(X_t) + \mathbf{I} - \Phi(X_t)^\top \Lambda_0^{-1} \Phi(X_c)(\Phi(X_c)^\top \Lambda_0^{-1} \Phi(X_c) + \mathbf{I})^{-1}(\Phi(X_c)^\top \Lambda_0^{-1} \Phi(X_t))] \\ &\otimes \Sigma_\epsilon\} \end{aligned} \tag{13}$$

2.3 Equivalence between GPR with Linear Kernel and Latent Space BLR

Proposition 1. *The posterior predictive distribution of BLR (Equation 1) and GPR (Equation 13) evaluates to the same distribution.*

Proof. By invoking the Woodbury identity, we show the detailed proof in Appendix A. □

Note although they both lead to the same posterior distribution, BLR has significant computational advantage over GPR during inference when the number of conditioned context data is large, as the dimension of matrix which needs to be inversed in Equation 12 is $n_x \times n_x$ while for BLR in Equation 2 is $n_\phi \times n_\phi$, irrespective of the number of context points.

It is also worth mentioning the redundant parametrization of $\mathbf{K_0}$ and Λ_0 in Latent BLR:

Proposition 2. *Latent BLR defined by prior parameters $\mathbf{K_0}, \Lambda_0$ and $\phi(x)$ leads to the same posterior predictive distribution (Equation 1) as Latent BLR defined by $\mathbf{K_0'}, \Lambda_0'$, and $\phi'(x)$ given by*

$$\mathbf{K_0'} = \mathbf{L}'^{\top}\mathbf{K_0} \tag{14}$$

$$\Lambda_0'^{-1} = \mathbf{L}'^{\top}\Lambda_0^{-1}\mathbf{L}' \tag{15}$$

$$\phi'(x) = \mathbf{L}'^{-1}\phi(x) \tag{16}$$

where $\mathbf{L}'$ is any invertible matrix.

Proof. See Appendix B. □

Therefore, $\mathbf{K_0}$ and Λ_0 are redundant in that one of the parameter could be fixed during training without an impact on the overall loss. Fixing Λ_0 during training has additionally led to better loss landscape and avoided numerical issues especially when used with a weight decay optimizer, which will be discussed in later sections.

Due to this equivalence, we can take $\mathbf{L}' = \mathbf{L}^{-1}$ which results in a $\Lambda_0' = \Lambda_0'^{-1} = \mathbf{I}$, rendering its equivalent GPR kernel a standard linear kernel $k'(x, x') = \phi'(x)^{\top}\phi'(x)$ instead of a weighted one.

2.4 Extended Equivalence to Mercel Kernel GPRs

It is a classical result in machine learning that Bayesian Linear Regression can be regarded as a special case of Gaussian Process Regression with which linear kernel adopted. We have shown in section 2.3 the two methods yield the same posterior distribution even for multi-variate case. In this section, we extend the argument of equivalence to all GPR models with any Mercer kernels.

Mercer kernels is the family of kernels whose gram matrix for any set of inputs (Equation 5) is always symmetric positive definite. Since all Mercer kernels can be written as the inner product in feature space: $k(x, x') = \phi(x)^{\top}\phi(x')$ Murphy [2013], we can show any GPR with a Mercer kernel and a linear mean function: $m(x) = \mathbf{K}^{\top}\phi(x)$ is equivalent to a BLR in the latent space defined by the feature transformation: $\phi(x)$ with $\mathbf{K_0} = \mathbf{K}$, and $\Lambda_0 = \mathbf{I}$.

Although not all kernels belong to the Mercer kernel family (such as the sigmoid kernel defined as $k(x, x') = \tanh(\gamma x^{\top}x' + r)$), many of the commonly used ones such as Gaussian kernel (RBF Kernel), linear kernel, Matern kernel are Mercer (Scholkopf and Smola [2001]). One caveat is that some kernels such as Gaussian kernel could have an infinite-dimensional corresponding feature vector $\phi(x)$ although its representation power could be approximated by a kernel with a large enough finite-dimensional feature vector. We show the empirical results in later sections.

2.5 Loss Function Equivalence

Apart from the architectural difference discussed above, the other disparity between ALPaCA and GPR Meta Learning outlined in Harrison et al. [2018] and Rothfuss et al. [2020] is the different cost functions which are used to optimize for prior parameters. ALPaCA assumes new data flow in a streaming fashion in that we observe one pair (x_t, y_t) before we observe the next pair (x_{t+1}, y_{t+1}). In this section, we show that considering this "per time-step" loss is equivalent to maximizing the overall likelihood of the meta dataset when assuming a uniform time distribution even under non-i.i.d. assumption of the underlying data.

Proposition 3. *For all probabilistic inference algorithms, the minimization of negative expected posterior likelihood (Equation 17 used by ALPaCA):*

$$\min_{\xi} \mathbb{E}_{t\sim p(t),\theta^*\sim p(\theta)}[\mathbb{E}_{x_{t+1},y_{t+1},D_t^*\sim p(x_{t+1},y_{t+1},D_t^*|\theta^*)} \log q_{\xi}(y_{t+1}|x_{t+1}, D_t^*)] \tag{17}$$

and expected negative prior likelihood over the entire horizon (Equation 18 used by PACOH-MAP):

$$\min_{\xi} \mathbb{E}_{\theta^*\sim p(\theta)}[\mathbb{E}_{D_T^*\sim p(D^*|\theta^*)} \log l_{\xi}(D_T^*)] \tag{18}$$

is equivalent, assuming a uniform distribution on the context data horizon t.

Proof. ALPaCA optimizes for the expected posterior likelihood as follows:

$$\min_{\xi=(\mathbf{K_0},\Lambda_0,\omega)} \mathbb{E}_{t\sim p(t),\theta^*\sim p(\theta)}[\mathbb{E}_{x_{t+1},y_{t+1},D_t^*\sim p(x_{t+1},y_{t+1},D_t^*|\theta^*)} \log q_\xi(y_{t+1}|x_{t+1},D_t^*)] \tag{19}$$

where q_ξ refers to the conditional probability distribution defined in Equation 1.

GPR MLL (Maximum Likelihood) directly optimizes for the likelihood of data over a horizon of time T across tasks without explicitly considering the distribution over time step:

$$\min_\xi \mathbb{E}_{\theta^*\sim p(\theta)}[\mathbb{E}_{D_T^*\sim p(D^*|\theta^*)} \log l_\xi(D_T^*)] \tag{20}$$

where l_ξ refers to the likelihood probability distribution defined in Equation 7.

Assuming uniform distribution on $p(t)$ for Equation 17 and a time horizon of T, it becomes:

$$\min_\xi \mathbb{E}_{\theta^*\sim p(\theta)} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}_{x_{t+1},y_{t+1},D_t^*\sim p(x,y,D_t^*|\theta^*)} \log q_\xi(y_{t+1}|x_{t+1},D_t^*) \tag{21}$$

$$\begin{aligned}
=&\frac{1}{T}\min_\xi \mathbb{E}_{\theta^*\sim p(\theta)}[\mathbb{E}_{x_1,y_1\sim p(x,y|\theta^*)} \log q_\xi(y_1|x_1)+\\
&\qquad \mathbb{E}_{x_2,y_2,x_1,y_1\sim p(x_2,y_2,D_1^*=(x_1,y_1)|\theta^*)} \log q_\xi(y_2|x_2,x_1,y_1)+\\
&\qquad \mathbb{E}_{x_3,y3,x_2,y_2,x_1,y_1\sim p(x_3,y_3,D_2^*=(x_1,y_1,x_2,y_2)|\theta^*)} \log q_\xi(y_3|x_3,x_1,y_1,x_2,y_2)+\\
&\qquad \ldots\ldots]
\end{aligned} \tag{22}$$

Note that the first term in Equation 22: $\mathbb{E}_{x_1,y_1\sim p(x,y|\theta^*)} \log q_\xi(y_1|x_1) = \mathbb{E}_{D_1^*\sim p(D^*|\theta^*)} \log l_\xi(D_1^*)$ since the posterior distribution q_ξ has not been conditioned on any context data.

We expand the first two terms in Equation 22 (note $p(x,y|\theta^*) = p(x|\theta^*)p(y|x,\theta^*)$, and purely for notational brevity, we drop the conditional dependence of x, y on θ^* below):

$$\begin{aligned}
&\mathbb{E}_{x_1,y_1\sim p(x,y|\theta^*)} \log q_\xi(y_1|x_1) + \underbrace{\mathbb{E}_{x_2,y_2,x_1,y_1\sim p(x_2,y_2,D_1^*=(x_1,y_1)|\theta^*)} \log q_\xi(y_2|x_2,x_1,y_1)}_{\mathbb{E}_2}\\
=&\int_{y_1}\int_{x_1} \log q_\xi(y_1|x_1)p(x_1)p(y_1|x_1)dx_1dy_1 + \mathbb{E}_2\\
=&\int_{y_1}\int_{x_1} \log q_\xi(y_1|x_1)p(x_1)p(y_1|x_1)\underbrace{\left(\int_{y_2}\int_{x_2} p(x_2|x_1)p(y_2|x_2)dx_2dy_2\right)}_{=1}dx_1dy_1 + \mathbb{E}_2\\
=&\int_{y_1}\int_{x_1}\int_{y_2}\int_{x_2} \log q_\xi(y_1|x_1)p(x_1)p(y_1|x_1)p(x_2|x_1)p(y_2|x_2)dx_2dy_2dx_1dy_1 + \mathbb{E}_2\\
=&\int_{x_2}\int_{y_2}\int_{x_1}\int_{y_1} \log \underbrace{q_\xi(y_1|x_1,x_2)}_{=q_\xi(y_1|x_1)} p(x_1)p(y_1|x_1)p(x_2|x_1)p(y_2|x_2)dy_1dx_1dy_2dx_2+\\
&\int_{x_2}\int_{y_2}\int_{x_1}\int_{y_1} \log q_\xi(y_2|x_2,x_1,y_1)p(x_1)p(y_1|x_1)p(x_2|x_1)p(y_2|x_2)dy_1dx_1dy_2dx_2\\
=&\int_{x_2}\int_{y_2}\int_{x_1}\int_{y_1} \log\left(q_\xi(y_1|x_1,x_2)q_\xi(y_2|x_2,x_1,y_1)\right)p(x_1,x_2)p(y_1|x_1)p(y_2|x_2)dy_1dx_1dy_2dx_2\\
=&\int_{x_2}\int_{y_2}\int_{x_1}\int_{y_1} \log\left(q_\xi(y_1,y_2|x_1,x_2)\right)p(x_1,x_2)p(y_1,y_2|x_1,x_2)dy_1dx_1dy_2dx_2\\
=&\int_{D_2^*} \log l_\xi(D_2^*)p(D_2^*|\theta^*)dD_2^* = \mathbb{E}_{D_2^*\sim p(D^*|\theta^*)} \log l_\xi(D_2^*)
\end{aligned} \tag{23}$$

Similarly by induction, we con conclude Equation 22 evaluates to

$$\frac{1}{T}\min_\xi \mathbb{E}_{\theta^*\sim p(\theta)}[\mathbb{E}_{D_T^*\sim p(D^*|\theta^*)} \log l_\xi(D_T^*)] \tag{24}$$

which is equivalent to the loss function (Equation 18) from GPR maximum likelihood method. □

3 Empirical Studies

In this section, we evaluate the performance of the above-mentioned Bayesian meta-learning frameworks with varying architectural and loss function differences.

3.1 Environment Setup

The source code used in the evaluations are based on implementations from Harrison et al. [2018] [3] and Rothfuss et al. [2020] [4].

Two synthetic dataset (Sinusoid, Cauchy) and one real-world dataset (Swissfel) are used for this benchmark. For sinusoid dataset, we generate training and test data from a family of sinusoid functions with added Gaussian noise: $y(x) = kx + A\sin(\omega x + b) + c + \epsilon$ with varying $k, A, \omega, b, c, \epsilon$ with their distribution summarized in Table 5. Cauchy dataset is generated via the superposition of a common mean function (mixture of two Gaussians) across all tasks with GP prior functions sampled with an SE kernel. This synthetic dataset is of particular interest for evaluating the advantage of having a separate mean function estimator over sharing the same representation for both mean and kernel modules. Task data in Swissfel dataset are collected from multiple calibration sessions of Swiss Free Electron Laser (SwissFEL) (Milne et al. [2017] Kirschner et al. [2019]). This dataset poses additional challenges such as potential overfitting due to the small amount of training tasks and samples available.

In this meta-learning setting, data from multiple tasks are included in the training dataset while test dataset consists of additional data from other unseen tasks. All data in training sets are used for optimizing bayesian prior hyper-parameters. Data in test sets are split into context data which are used to perform online update in BLR or GPR and test data which are evaluated for performance. To ensure a fair comparison between experiments, methods are trained and evaluated on the same training set and test set. Table 4 summarizes the dataset configurations.

For each experiment, AdamW (Loshchilov and Hutter [2019]) optimizer are chosen over Adam (Kingma and Ba [2014]) to avoid overfitting during training. The parameters of the optimizer such as learning rate and weight decay are tuned to achieve convergence in both training loss and validation loss. Lastly, we selected the best parameters based on performance in validation set (maximum log likelihood w.r.t posterior predictive distribution).

We evaluate the methods on three metrics: log likelihood of posterior predictive distribution, RMSE (root mean square error) and calibration error Kuleshov et al. [2018] on test set. While RMSE captures the accuracy of mean prediction, calibration error reflects the accuracy in uncertainty estimates, namely the RMSE between empirical frequency and predicted frequency across different confidence intervals around mean prediction.

3.2 Architecture and Loss Function

In this section, we investigate the effect of different learning architecture and objective functions. The different configurations used for comparison can be found in Table 1. For GP-based methods, we investigate the use of different kernel and mean function choices. 'SE' refers to the use of a Squared Exponential Kernel directly on original data. 'DSE' and 'DL' correspond to the use of a Squared Exponential or Linear Kernel on a latent representation of raw data transformed by a deep neural network. An independent neural network is used for mean function prediction for 'IN' models while the mean function in 'SN' cases shares the same latent representation with its kernel.

[3] ALPaCA: `https://github.com/StanfordASL/ALPaCA`

[4] PACOH: `https://github.com/jonasrothfuss/meta_learning_pacoh`

Abbrev.	Architecture	Loss	Reference
GPR-SE-IN	GPR w/ SE kernel, independent deep mean function	Equation 28: prior likelihood w/ full covariance	
GPR-DSE-IN	GPR w/ deep SE kernel, independent deep mean function	Equation 28: prior likelihood w/ full covariance	Rothfuss et al. [2020] (MAP)
GPR-DL-IN	GPR w/ deep linear kernel, independent deep mean function	Equation 28: prior likelihood w/ full covariance	
GPR-DL-SN	GPR w/ deep linear kernel, shared deep mean function	Equation 28: prior likelihood w/ full covariance	
BLR-PR-FC	BLR	Equation 28: prior likelihood w/ full covariance	
BLR-PR-DC	BLR	Equation 29: prior likelihood w/ diagonal covariance	
BLR-POO-D/FC	BLR	Equation 25: posterior likelihood on (x_{t+1}, y_{t+1})	Harrison et al. [2018]
BLR-POM-FC	BLR	Equation 27: posterior likelihood w/ full covariance on $(x_{t+1}, y_{t+1}, \dots, x_T, y_T)$	
BLR-POM-DC	BLR	Equation 26: posterior likelihood w/ diagonal covariance on $(x_{t+1}, y_{t+1}, \dots, x_T, y_T)$	ALPaCA Code

Table 1: Model Abbreviations and configuration details.

While the architecture choice for BLR is limited (equivalent to GPR-DL-SN as shown in Proposition 1), the equivalence between prior and posterior loss function (Proposition 3) is examined empirically.

Note in the original ALPaCA paper, it is stated that the objective (posterior log likelihood) is evaluated on the next available data point (Equation 17):

$$\mathbb{E}_{t\sim p(t),\theta^*\sim p(\theta)}\{\mathbb{E}_{x_{t+1},y_{t+1},D_t^*\sim p(x_{t+1},y_{t+1},D_t^*|\theta^*)} \log q_\xi(y_{t+1}|x_{t+1}, D_t^*)\} \tag{25}$$

while the paper's accompanying source code implements it as:

$$\mathbb{E}_{t\sim p(t),\theta^*\sim p(\theta)}\{\mathbb{E}_{\tau\sim\mathcal{U}(t+1,T)}[\mathbb{E}_{x_\tau,y_\tau,D_t^*\sim p(x_{t+1},y_{t+1},D_t^*|\theta^*)} \log q_\xi(y_\tau|x_\tau, D_t^*)]\} \tag{26}$$

We propose a similar version which also takes into account the covariance between data points as follows:

$$\mathbb{E}_{t\sim p(t),\theta^*\sim p(\theta)}\{\mathbb{E}_{X,Y,D_t^*\sim p(x_{t+1,\dots,T},y_{t+1,\dots,T},D_t^*|\theta^*)} \log q_\xi(Y|X, D_t^*)\} \tag{27}$$

Similarly, for the prior likelihood cost (Equation 18) evaluated on a batch of data D^*:

$$\mathbb{E}_{\theta^*\sim p(\theta)}[\mathbb{E}_{D_T^*\sim p(D^*|\theta^*)} \log l_\xi(D_T^*)] \tag{28}$$

we can derive the version evaluated on a single sample averaged over the dataset as follows:

$$\mathbb{E}_{\theta^*\sim p(\theta)}\{\mathbb{E}_{\tau\sim\mathcal{U}(0,T)}[\mathbb{E}_{x_\tau,y_\tau\sim p(x_\tau,y_\tau|\theta^*)} \log l_\xi(x_\tau, y_\tau)]\} \tag{29}$$

Starting from the model in PACOH-MAP(Rothfuss et al. [2020]), by changing one element at a time, we generated a total of 8 models with different architecture and loss function combination before we arrive at the standard ALPaCA (Harrison et al. [2018]) implementation. The configurations of these models are summarized in Table 1. The performance of these methods across datasets are presented in Table 2 and Table 3.

Method	Sinusoid-Easy	Sinusoid-Hard	Cauchy	Swissfel
GPR-SE-IN	0.313	-0.112	0.394	-0.447
GPR-DSE-IN	0.596	-0.348	0.185	0.763
GPR-DL-IN	0.122	-0.768	-0.015	-1.228
GPR-DL-SN	0.141	-0.793	0.016	-0.645
BLR-PR-FC	-0.203	-0.362	0.011	-0.826
BLR-PR-DC	-1.210	-1.707	-0.308	-1.768
BLR-POO-D/FC	-0.450	-0.256	0.044	-0.979
BLR-POM-FC	-0.373	-0.379	-0.038	-1.892
BLR-POM-DC	-0.587	-0.401	-0.193	-1.406

Table 2: Comparison of Log Likelihood of GPR vs BLR meta-learning methods with different configurations across different datasets.

	Sinusoid-Easy		Sinusoid-Hard		Cauchy		Swissfel	
Method	RMSE	Calib.	RMSE	Calib.	RMSE	Calib.	RMSE	Calib.
GPR-SE-IN	0.315	0.120	0.644	0.108	0.200	0.060	0.368	0.086
GPR-DSE-IN	0.287	0.124	0.614	0.104	0.217	0.069	0.443	0.057
GPR-DL-IN	0.248	0.130	0.637	0.105	0.239	0.074	0.663	0.076
GPR-DL-SN	0.218	0.142	0.644	0.109	0.230	0.076	0.459	0.054
BLR-PR-FC	0.340	0.118	0.591	0.097	0.225	0.078	0.479	0.074
BLR-PR-DC	0.748	0.173	0.878	0.147	0.237	0.112	0.641	0.146
BLR-POO-D/FC	0.438	0.111	0.643	0.100	0.231	0.075	0.630	0.078
BLR-POM-FC	0.404	0.116	0.627	0.104	0.246	0.080	0.828	0.139
BLR-POM-DC	0.481	0.132	0.725	0.104	0.234	0.102	0.967	0.143

Table 3: Comparison of RMSE and Calibration Error GPR vs BLR meta-learning methods with different configurations across different datasets.

Architecture Equivalence As discussed in Proposition 1, subjected to the same loss function, GPR-DL-SN and BLR-PR-FC in theory should yield the same posterior predictive distribution. Although GPR-DL-SN achieves slightly better performance in Sinusoid-Easy, we observe comparable results on other three datasets. The disparity in performance is most likely due to different optimization outcomes rather than representation power difference. Further benchmark on more dataset is required to determine whether one architecture has the advantage over the other from an optimization point of view.

Prior and Posterior Loss Equivalence It is argued in Proposition 3 that PR-FC is mathematically equivalent to POO-D/FC. Across all four datasets, we observe a small advantage in RMSE of BLR-PR-FC over BLR-POO-D/FC. BLR-PR-FC and BLR-POO-D/FC are also consistently the best two performed models among the five loss choices in terms of log likelihood.

Full vs Diagonal Covariance Loss A significant drop in log likelihood is noticed after switching from PR-FC/POM-FC to PR-DC/POM-DC across all dataset, with the exception of Swissfel where both POM-FC and POM-DC fails. This suggests that capturing the correlation between data points in the loss function is beneficial to ensure good generalization, which is an important aspect of prior learning.

Posterior on One vs All Samples BLR-POO-D/FC and BLR-POM-FC performs similarly except for Swissfel dataset. For i.i.d data with $p(x_t, y_t) = p(x_{t'}, y_{t'})$, Equation 25 and Equation 27 evaluates to the same quantity. Since the data in Swissfel corresponds to real data collected during a a calibration process which uses Bayesian optimization (Kirschner et al. [2019]), the non-i.i.d property of the training data may have attributed to this drop.

Shared vs Independent Latent Representation between Linear Kernel and Mean Function In theory, having a separate neural network for mean function prediction should be more expressive and leads to better performance than the case with shared representation for both kernel feature space and mean function. However, sharing the same latent space might force it to learn a better representation which is meaningful both in terms of target variable and correlation between samples. Having a separate mean deep network did not yield improvement in Sinusoid Dataset, Cauchy Dataset and has

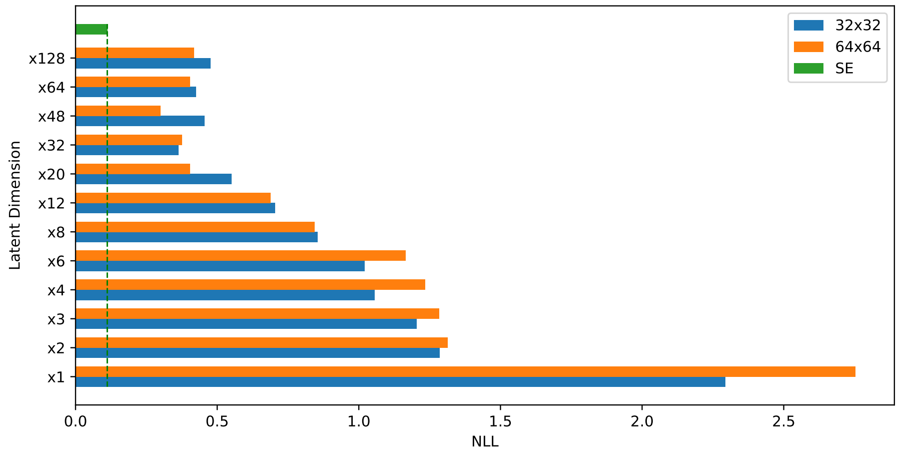

Figure 1: NLL(Negative Log Likelihood) on Sinusoid-Hard test set by BLR-PR-FC with varying latent space $\phi(x)$ size. As an example, the bottom orange bar corresponds to a $\phi(x)$ parameterized by a 64x64x1 network.

led to poorer performance in Swissfel dataset. The benefit of having an independent mean module needs to be analyzed on a per-case basis.

3.3 Approximation of Mercer Kernel with Neural Networks

In section 2.4, we argue that any Mercer Kernel can be approximated by a linear kernel operating on a finite-dimensional latent space. To study the influence of network size of this latent representation on the quality of covariance prediction, we experiment BLR-PR-FC with different network sizes on Sinusoid-Hard dataset.

As shown in Figure 1, while widening the feature dimension of the latent space, there is an initial trend in performance increase until the dimension reaches 32, after which the performance plateaus. We can observe a shift of bottleneck from the representation power of latent representation to the insufficient data available for further exploitation. The initial improvement is also reflected in the posterior predictive distribution shown in Figure 2 and Figure 3.

Empirically, it is apparent this learned latent space representation is not as effective as a common kernel such as SE kernel. Among all candidates, GPR-SE-IN and GPR-DSE-IN are the best performing models across all dataset with significant margins compared to GPR-DL-IN as shown in Table 2.

In conclusion, it is helpful to adopt a hand-crafted kernel, which already measures meaningful similarity, either on raw data or transformed latent space, if necessary. It is highly unlikely, with the limited amount of training data, that a latent feature representation of a powerful kernel could be directly learned.

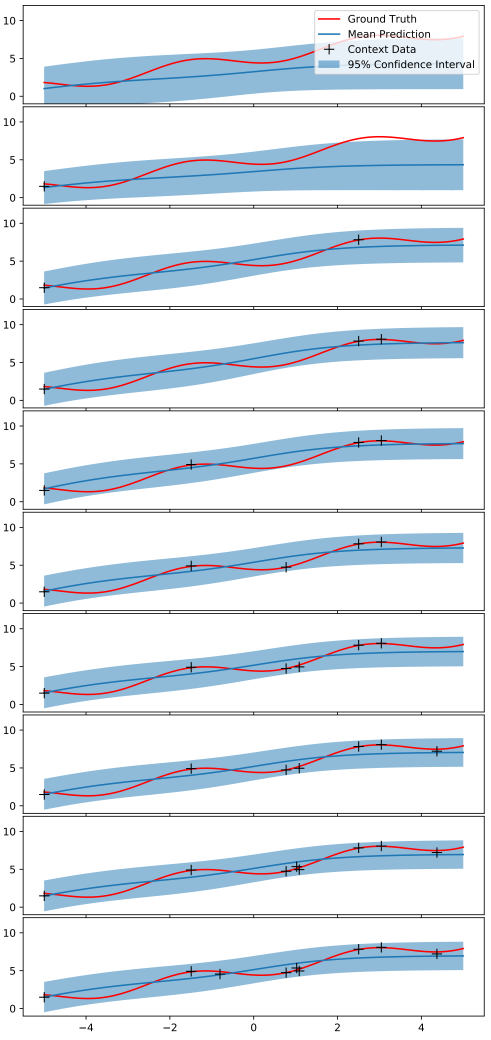

Figure 2: Posterior Prediction by BLR-PR-FC with $\phi(x)$ of size 32x32x3 on Sinusoid-Hard test set.

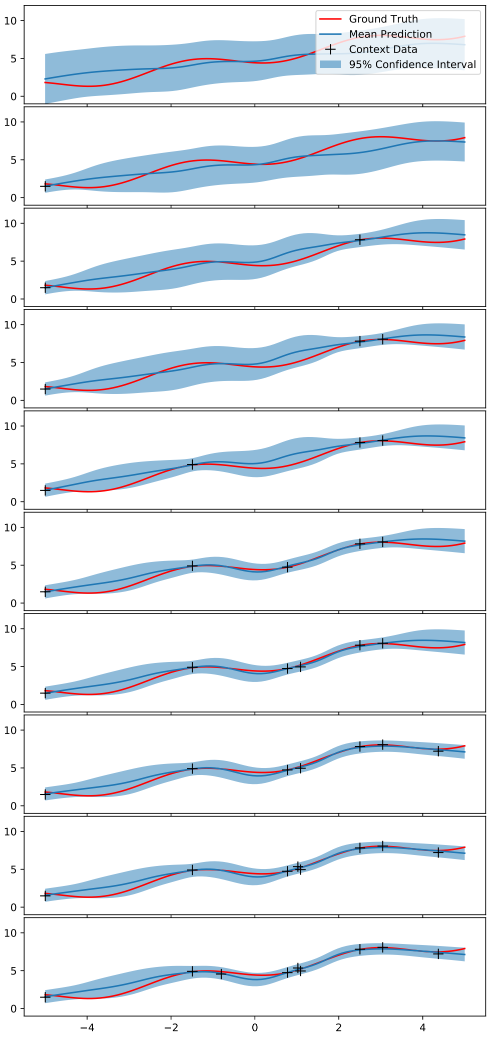

Figure 3: Posterior Prediction by BLR-PR-FC with $\phi(x)$ of size 32x32x32 on Sinusoid-Hard test set.

4 Conclusion

In this report, we compared ALPaCA vs. PACOH-MAP, two recently published bayesian meta-learning frameworks and their variations. We showed theoretical equivalence in their model architecture, namely the equivalence between BLR (Bayesian Linear Regression) and GPR (Gaussian Process Regression). We extended this equivalence to the range of GPs with any Mercer kernel. We further showed the objective function used in ALPaCA and PACOH are also equivalent under mild assumptions. Further empirical studies verified this architectural and loss function equivalence. Furthermore, other model derivatives with similar architecture and loss function revealed the importance of considering full covariance when evaluating multiple samples and the benefit of adopting an 'already-learned' kernel in boosting performance with limited data.

We hope to continue investigating the comparison in the reinforcement learning setting where both training and test data are inherently non i.i.d. We believe it is also of interest to explore whether kernel approximation could be separated from the meta-learning process in that we first train a neural network to approximate a common kernel and use the fixed network in BLR to achieve fast update without compromise on performance.

References

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, Mar 2000. ISSN 1076-9757. doi: 10.1613/jair.731. URL `http://dx.doi.org/10.1613/jair.731`.

Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees, 2017.

Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold gaussian processes for regression. *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul 2016. doi: 10.1109/ijcnn.2016.7727626. URL `http://dx.doi.org/10.1109/IJCNN.2016.7727626`.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Rachit Dubey, Pulkit Agrawal, Deepak Pathak, Thomas L. Griffiths, and Alexei A. Efros. Investigating human priors for playing video games, 2018.

Vincent Fortuin, Heiko Strathmann, and Gunnar Rätsch. Meta-learning mean functions for gaussian processes, 2019.

Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, and S. M. Ali Eslami. Conditional neural processes, 2018.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes, 2018.

James Harrison, Apoorva Sharma, and Marco Pavone. Meta-learning priors for efficient online bayesian regression. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018.

Lukas Hewing, Juraj Kabzan, and Melanie N. Zeilinger. Cautious model predictive control using gaussian process regression. *IEEE Transactions on Control Systems Technology*, pages 1–8, 2019. ISSN 2374-0159. doi: 10.1109/tcst.2019.2949757. URL `http://dx.doi.org/10.1109/TCST.2019.2949757`.

Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proceedings in 3rd International Conference for Learning Representations*, 2014.

Johannes Kirschner, Manuel Nonnenmacher, Mojmir Mutnỳ, Andreas Krause, Nicole Hiller, Rasmus Ischebeck, and Andreas Adelmann. Bayesian optimisation for fast and safe parameter tuning of swissfel. In *FEL2019, Proceedings of the 39th International Free-Electron Laser Conference*, pages 707–710. JACoW Publishing, 2019.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression, 2018.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings in 5th International Conference for Learning Representations*, 2019.

Christopher J Milne, Thomas Schietinger, Masamitsu Aiba, Arturo Alarcon, Jürgen Alex, Alexander Anghel, Vladimir Arsov, Carl Beard, Paul Beaud, Simona Bettoni, et al. Swissfel: the swiss x-ray free electron laser. *Applied Sciences*, 7(7):720, 2017.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen. King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Kevin P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.], 2013. ISBN 9780262018029 0262018020.

Valentin Peretroukhin, Lee Clement, and Jonathan Kelly. Reducing drift in visual odometry by inferring sun direction using a bayesian convolutional neural network. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'17)*, Singapore, May 29–Jun. 3 2017.

Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rkgpy3C5tX`.

Jonas Rothfuss, Vincent Fortuin, and Andreas Krause. Pacoh: Bayes-optimal meta-learning with pac-guarantees, 2020.

Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001. ISBN 0262194759.

Joaquin Vanschoren. Meta-learning: A survey, 2018.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning, 2015.

A Proof of Equivalence on Posterior Distribution of GPR and BLR

In this section, we prove that the distribution of Equation 1 and Equation 13 are the same.

For the Gaussian distribution mean, starting from Equation 13:

$$\begin{aligned}
\mu_{GPR} =& \Phi(X_t)^\top \mathbf{K_0} + \Phi(X_t)^\top \Lambda_0^{-1}\Phi(X_c)(\Phi(X_c)^\top\Lambda_0^{-1}\Phi(X_c)+\mathbf{I})^{-1}(Y_c - \Phi(X_c)^\top\mathbf{K_0} && (30)\\
=& \Phi(X_t)^\top[\mathbf{K_0} + \Lambda_0^{-1}\Phi(X_c)(\Phi(X_c)^\top\Lambda_0^{-1}\Phi(X_c)+\mathbf{I})^{-1}(Y_c - \Phi(X_c)^\top\mathbf{K_0})] && (31)\\
=& \Phi(X_t)^\top[\mathbf{I} - \Lambda_0^{-1}\Phi(X_c)(\Phi(X_c)^\top\Lambda_0^{-1}\Phi(X_c)+\mathbf{I})^{-1}\Phi(X_c)^\top]\mathbf{K_0}\\
&+ \Phi(X_t)^\top[\Lambda_0^{-1}\Phi(X_c)(\Phi(X_c)^\top\Lambda_0^{-1}\Phi(X_c)+\mathbf{I})^{-1}]Y_c && (32)
\end{aligned}$$

Note that by applying Woodbury identity [5] on Equation 3:

$$\begin{aligned}
\Lambda_\tau^{-1} &= (\Lambda_0 + \Phi(X_c)\mathbf{I}\Phi(X_c)^\top)^{-1}\\
&= \Lambda_0^{-1} - \Lambda_0^{-1}\Phi(X_c)(\mathbf{I}+\Phi(X_c)^\top\Lambda_0^{-1}\Phi(X_c))^{-1}\Phi(X_c)^\top\Lambda_0^{-1} && (33)\\
\Lambda_\tau^{-1}\Lambda_0 &= \mathbf{I} - \Lambda_0^{-1}\Phi(X_c)(\mathbf{I}+\Phi(X_c)^\top\Lambda_0^{-1}\Phi(X_c))^{-1}\Phi(X_c)^\top && (34)
\end{aligned}$$

In addition,

$$\begin{aligned}
\Lambda_\tau\Lambda_0^{-1}\Phi(X_c) &= (\Phi(X_c)\Phi(X_c)^\top + \Lambda_0)\Lambda_0^{-1}\Phi(X_c) && (35)\\
&= \Phi(X_c)\Phi(X_c)^\top\Lambda_0^{-1}\Phi(X_c) + \Phi(X_c) && (36)\\
&= \Phi(X_c)(\mathbf{I}+\Phi(X_c)^\top\Lambda_0^{-1}\Phi(X_c)) && (37)
\end{aligned}$$

Multiply both sides by Λ_τ^{-1},

$$\begin{aligned}
\Lambda_0^{-1}\Phi(X_c) &= \Lambda_\tau^{-1}\Phi(X_c)(\mathbf{I}+\Phi(X_c)^\top\Lambda_0^{-1}\Phi(X_c)) && (38)\\
\Lambda_0^{-1}\Phi(X_c)(\mathbf{I}+\Phi(X_c)^\top\Lambda_0^{-1}\Phi(X_c))^{-1} &= \Lambda_\tau^{-1}\Phi(X_c) && (39)
\end{aligned}$$

Substitute Equation 34 and Equation 39 into Equation 32,

$$\begin{aligned}
\mu_{GPR} &= \Phi(X_t)^\top\Lambda_\tau^{-1}\Lambda_0\mathbf{K_0} + \Phi(X_t)^\top\Lambda_\tau^{-1}\Phi(X_c)Y_c && (40)\\
&= \Phi(X_t)^\top\Lambda_\tau^{-1}(\Lambda_0\mathbf{K_0} + \Phi(X_c)Y_c) && (41)\\
&= \mu_{BLR} && (42)
\end{aligned}$$

[5] $(\mathbf{A}+\mathbf{CBC}^\top)^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{C}(\mathbf{B}^{-1}+\mathbf{C}^\top\mathbf{A}^{-1}\mathbf{C})^{-1}\mathbf{C}^\top\mathbf{A}^{-1}$

For the covariance matrix, we start from Equation 13:

$$\begin{aligned}
&\Sigma_{GPR} \\
=&[\Phi(X_t)^\top \Lambda_0^{-1}\Phi(X_t) + \mathbf{I} - \Phi(X_t)^\top \Lambda_0^{-1}\Phi(X_c)(\Phi(X_c)^\top \Lambda_0^{-1}\Phi(X_c)+\mathbf{I})^{-1}(\Phi(X_c)^\top \Lambda_0^{-1}\Phi(X_t))] \\
&\otimes \Sigma_\epsilon \\
=&\{\mathbf{I} + \Phi(X_t)^\top [\Lambda_0^{-1} - \Lambda_0^{-1}\Phi(X_c)(\Phi(X_c)^\top \Lambda_0^{-1}\Phi(X_c)+\mathbf{I})^{-1}\Phi(X_c)^\top \Lambda_0^{-1}]\Phi(X_t)\} \otimes \Sigma_\epsilon \quad (43)
\end{aligned}$$

Subsitute Equation 33, we get:

$$\Sigma_{GPR} = (\mathbf{I} + \Phi(X_t)^\top \Lambda_\tau^{-1}\Phi(X_t)) \otimes \Sigma_\epsilon = \Sigma_{BLR} \quad (44)$$

B Proof of Equivalent Posterior Distribution with Linearly Transformed $\mathbb{K}$ Prior Distribution under Latent Bayesian Linear Regression

Since both Λ_0 and Λ_0^{-1} are symmetric positive-definite matrices, we can perform Cholesky decomposition on them: $\Lambda_0^{-1} = \mathbf{L}^\top \mathbf{L}$.

We claim the posterior predictive distribution in Equation 1 is equivalent after a linear transformation on $\Lambda_0'^{-1} = \mathbf{L}'^\top \Lambda_0^{-1}\mathbf{L}'$, $\phi'(x) = \mathbf{L}'^{-1}\phi(x)$ and $\mathbf{K_0'} = \mathbf{L}'^\top \mathbf{K_0}$:

$$\begin{aligned}
p(Y_t|X_t, X_c, Y_c) &= \mathcal{N}(\Phi(X_t)^\top \mathbf{K}_\tau, \Sigma_\tau) \\
&= \mathcal{N}(\Phi'(X_t)^\top \mathbf{K}_\tau', \Sigma_\tau')
\end{aligned}$$

First note,

$$\begin{aligned}
\Phi'(X_t)^\top &= \Phi(X_t)^\top \mathbf{L}'^{-\top} && (45)\\
\Lambda_\tau'^{-1} &= (\Phi'(X_c)\Phi'(X_c)^\top + \Lambda_0')^{-1} && (46)\\
&= (\mathbf{L}'^{-1}\Phi(X_c)\Phi(X_c)^\top \mathbf{L}'^{-\top} + \mathbf{L}'^{-1}\Lambda_0\mathbf{L}'^{-\top})^{-1} && (47)\\
&= \left[\mathbf{L}'^{-1}(\Phi(X_c)\Phi(X_c)^\top + \Lambda_0)\mathbf{L}'^{-\top}\right]^{-1} = \left[\mathbf{L}'^{-1}\Lambda_\tau\mathbf{L}'^{-\top}\right]^{-1} && (48)\\
&= \mathbf{L}'^\top \Lambda_\tau^{-1}\mathbf{L}' && (49)\\
\mathbf{K}_\tau' &= \Lambda_\tau'^{-1}(\Phi'(X_c)Y_c + \Lambda_0'\mathbf{K_0'}) && (50)\\
&= \Lambda_\tau'^{-1}(\mathbf{L}'^{-1}\Phi(X_c)Y_c + \mathbf{L}'^{-1}\Lambda_0\mathbf{L}'^{-\top}\mathbf{L}'^\top \mathbf{K_0}) && (51)\\
&= \mathbf{L}'^\top \Lambda_\tau^{-1}\mathbf{L}'\mathbf{L}'^{-1}(\Phi(X_c)Y_c + \Lambda_0\mathbf{K_0}) && (52)\\
&= \mathbf{L}'^\top \Lambda_\tau^{-1}(\Phi(X_c)Y_c + \Lambda_0\mathbf{K_0}) = \mathbf{L}'^\top \mathbf{K}_\tau && (53)
\end{aligned}$$

Apply the above for the distribution mean and covariance:

$$\begin{aligned}
\Phi'(X_t)^\top \mathbf{K}_\tau' &= \Phi(X_t)^\top \mathbf{L}'^{-\top}\mathbf{L}'^\top \mathbf{K}_\tau = \Phi(X_t)^\top \mathbf{K}_\tau && (54)\\
\Sigma_\tau' &= (\mathbf{I} + \Phi'(X_t)^\top \Lambda_\tau'^{-1}\Phi'(X_t)) \otimes \Sigma_\epsilon && (55)\\
&= (\mathbf{I} + \Phi(X_t)^\top \mathbf{L}'^{-\top}\mathbf{L}'^\top \Lambda_\tau^{-1}\mathbf{L}'\mathbf{L}'^{-1}\Phi(X_t)) \otimes \Sigma_\epsilon && (56)\\
&= (\mathbf{I} + \Phi(X_t)^\top \Lambda_\tau^{-1}\Phi(X_t)) \otimes \Sigma_\epsilon = \Sigma_\tau && (57)
\end{aligned}$$

C Dataset Details

The following hyper-parameters are used for generating and evaluation of the datasets:

C.1 Sinusoid-Easy and Sinusoid-Hard

Both dataset are generated by uniformly sampling the family of sinusoid functions: $y(x) = kx + A\sin[\omega(x-b)] + c + \epsilon$ from $X = [-5, 5]$. The probability distribution of $k, A, \omega, b, c, \epsilon$ are as follows:

C.2 Cauchy & Swissfel

Please refer to Appendix C.1.2. and C.1.3. in Rothfuss et al. [2020] for more information.

	Training Set		Test Set		
Dataset	# Training tasks	# Samples	# Test tasks	# Context samples	# Test samples
Sinusoid-Easy	20	5	100	5	100
Sinusoid-Hard	20	10	100	10	100
Cauchy	20	20	1000	20	100
Swissfel	5	200	4	200	200

Table 4: Hyper-parameters used in synthetic dataset generation and evaluation. Number of samples are per-task.

Parameter	Sinusoid-Easy	Sinusoid-Hard
k	$\mathcal{N}(0.5, 0.2^2)$	$\mathcal{N}(0.5, 0.6^2)$
A	$\mathcal{U}(0.7, 1.3)$	$\mathcal{U}(0.7, 1.4)$
ω	1.5	$\mathcal{U}(1.0, 2.0)$
b	$\mathcal{N}(0.1, 0.1^2)$	$\mathcal{N}(0.0, 2.0^2)$
c	$\mathcal{N}(5.0, 0.1^2)$	$\mathcal{N}(5.0, 0.8^2)$
ϵ	$\mathcal{N}(0.0, 0.1^2)$	$\mathcal{N}(0.0, 0.2^2)$

Table 5: Sinusoid synthetic datasets configuration parameters.

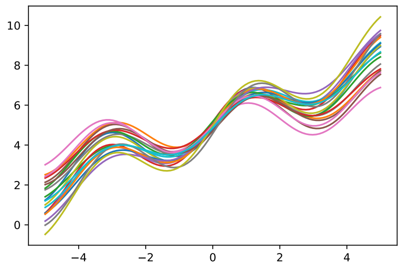

Figure 4: 20 test functions drawn from Sinusoid-Easy test set.

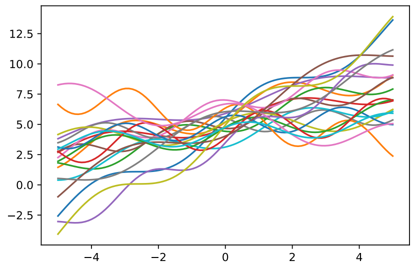

Figure 5: 20 test functions drawn from Sinusoid-Hard test set.