Universidad Autónoma de Madrid | Escuela Politécnica Superior

DEEP VARIATIONAL IMPLICIT PROCESS

LUIS ANTONIO ORTEGA ANDRÉS

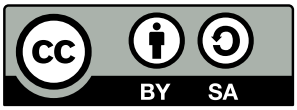
Master's Thesis

Data Science

Supervisors
Daniel Hernández Lobato
Simón Rodríguez Santana

SUPERIOR POLYTECHNIC SCHOOL
DEPARTMENT OF COMPUTER ENGINEERING
AUTONOMOUS UNIVERSITY OF MADRID

Madrid, Friday 10th June, 2022



The source code of this text and developed programs are available in the Github repository DeepVIP.

CONTENTS

INTRODUCTION 3

I BACKGROUND

1 BAYESIAN INFERENCE 8
- 1.1 Introduction . . . 8
- 1.2 Mean field family . . . 10
- 1.3 Black box α-energy . . . 11

2 GAUSSIAN PROCESSES 13
- 2.1 Motivation and definition . . . 13
- 2.2 Learning a Gaussian process . . . 14
- 2.3 Hyper-parameters learning . . . 16
- 2.4 Computational efficiency . . . 17
- 2.5 Inducing points approximation . . . 18
- 2.6 Bayesian linear regression approximation . . . 19

3 VARIATIONAL IMPLICIT PROCESSES 22
- 3.1 Model definition . . . 22
- 3.2 Computational complexity . . . 25
- 3.3 IPs generalize GPs . . . 25
- 3.4 Gaussian Kullback-Leibler . . . 25
- 3.5 Gaussian expectation . . . 26

4 RELATED RESEARCH 28

II DEEP VARIATIONAL IMPLICIT PROCESS

5 DEFINITION 31

6 MODEL DETAILS 34
- 6.1 Sampling from the marginal posterior approximation . . . 34
- 6.2 Making predictions for new instances . . . 34
- 6.3 Input propagation . . . 35
- 6.4 Computational complexity . . . 36

7 FURTHER DERIVATIONS 37

8 CONDUCTED EXPERIMENTS 39
- 8.1 Experimental settings . . . 39
- 8.2 Results . . . 40

9 IMPLEMENTATION DETAILS 47

CONCLUSIONS 48

ACKNOWLEDGMENTS 48

INTRODUCTION

Supervised learning problems seek to infer an unknown function that produces a certain output associated with each set of input attributes. To this end, a set of labeled data is needed, which consists on pairs of attributes and target values that enable optimization of the associated objective function. In recent years, models based on artificial neural networks have shown very good results in tackling this problem (Bergmann *et al.*, 2014; Anjos *et al.*, 2015).

Moreover, artificial neural networks have caught the interest of researchers in a wide range of domains, such as physics and biology (Baldi *et al.*, 2014; Anjos *et al.*, 2015) or computer vision and artificial intelligence (Kalchbrenner & Blunsom, 2013; Mnih *et al.*, 2013). However, in many fields of study, accurately reflecting the model's uncertainty is critical (Ghahramani, 2015), something ordinary neural networks can not accomplish.

For example, consider a model trained to detect if a tumor is carcinoid or not given a magnetic resonance imaging (MRI) of the patient (Hajjo *et al.*, 2021). Given a never before seen MRI structure, the desirable behaviour of the model in such instance would inform the user about the novelty of the input. The same reasoning applies to other fields such as models trained for autonomous driving cars, where the sensors detect a situation in the road that has not being previously considered. This desirable information of that the input is outside of the previously studied data can be interpreted as the level of uncertainty the model considers under its own prediction. In addition, this can be seen as the ability to know what the model does not know.

By using a Bayesian approach (MacKay, 1995), it is possible to capture the uncertainty associated with the predictions made by a neural network. In this approach, a prior distribution is specified for the model parameters that reflects our belief in the possible values these parameters can take before looking at the data. This prior distribution is combined with a likelihood, which measures how well each combination of parameters explains the observed data, to produce, using Bayes' rule, a posterior distribution over the set of parameters. This posterior distribution captures the value of the parameters that are compatible with the observed data. In practise, it is common that several parameter configuration do well in explaining the observed data (Figure 1).

Recently, the Bayesian approach has become popular for capturing the uncertainty associated to the predictions made by models that otherwise provide point-wise estimates, such as neural networks (NNs) (Gelman *et al.*, 2013; Gal, 2016; Murphy, 2012). However, when carrying out Bayesian inference, obtaining the posterior distribution in the space of parameters can become a limiting factor since it is often intractable. This is usually approached by the usage of approximated inference methods, which make is possible to learn models when exact inference is not applicable. These methods are commonly divided in sampling-based algorithms, such as Markov Chain Monte Carlo, which approximates the objective distribution using the stationary distribution of a specific Markov chain; and optimization-

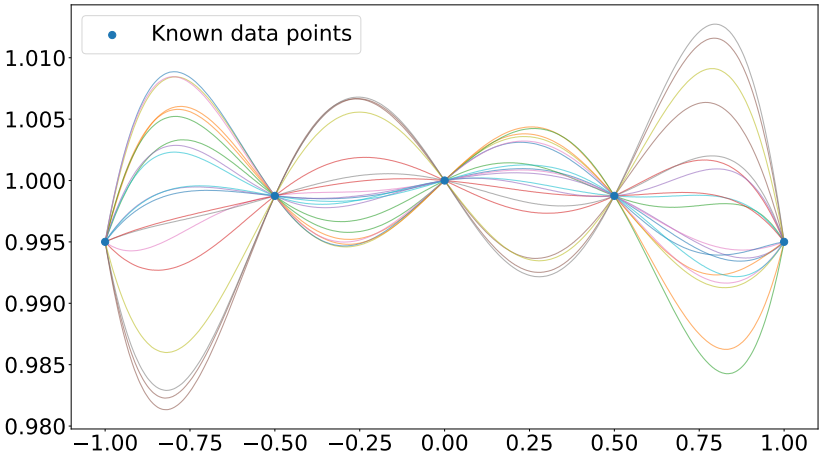


Figure 1: Functions defined by different parameter combinations that equally explain the observed (known) data points.

based algorithms, such as variational inference (VI) or expectation propagation (EP), which exchange the inference problem with an optimization one. However, the approximated inference problem is further complicated by the appearance of symmetries and strong dependencies between parameters. This is precisely the case in large deep NNs. Nevertheless, these issues can be alleviated by carrying out approximate inference in the space of functions, which presents certain advantages due to the simplified optimization space. This makes the approximations obtained in this space more precise than those obtained in parameter-space, as shown by recent works in the literature (Ma *et al.*, 2019; Sun *et al.*, 2019; Santana *et al.*, 2021; Ma & Hernández-Lobato, 2021).

A popular method carrying out function-space approximate inference are *Variational Implicit Processes* (VIPs) (Ma *et al.*, 2019). VIPs consider an implicit process (IP) as the prior distribution over the target latent function. IPs constitute a very flexible family of priors over functions that generalize Gaussian processes (GPs) (Ma *et al.*, 2019). Specifically, IPs are processes that may lack a closed-form expression, but that are easy-to-sample-from. Examples include Bayesian neural networks (BNN), neural samplers and wrapped GPs, among others (Santana *et al.*, 2021). Nevertheless, approximating the posterior of an IP-based prior is a challenging problem since it is also intractable most cases (except in the particular case of GPs). VIPs address this issue by approximating the posterior using the posterior of a GP with the same mean and covariances as the prior IP. Thus, the approximation used in VIPs results in a Gaussian predictive distribution, which may be too restrictive in real world scenarios.

Over the last years several works have employed the concatenation of random processes to increase the flexibility of predictive models which may initially be more limited. An important example here are *deep GPs* (DGPs) in which one GP is used as the input of another GP systematically (Damianou & Lawrence, 2013). Based on the success of the concatenation of GPs, it is natural to consider the concatenation of IPs to extend their capabilities in a similar fashion to DGPs. Therefore, *deep VIPs* (DVIPs) are introduced as a multi-layer extension of VIP that provides increased expressive power, enables more accurate predictions, gives better calibrated uncertainty estimates, and captures more complex patterns in the data. Figure 13 shows the architecture considered in DVIPs. Each layer contains several IPs that

are approximated using VIPs. Importantly, the flexibility of the IP-based prior formulation enables numerous models to be used as the prior over functions, leveraging the benefits of task-specific models, *e.g.*, convolutional NNs, which increase the performance on image datasets. Critically, DVIPs can adapt the prior IPs to the observed data, resulting in improved performance. When GP priors are considered, DVIPs are equivalent to DGPs. Therefore, it can be considered as a generalization of DGPs.

Approximate inference in DVIPs is carried out via variational inference. Computational scalability is achieved in each unit using a linear approximation of the GP that approximates the prior IP, as done in VIP (Ma *et al.*, 2019). The predictive distribution of such a model is Gaussian. However, since the inputs in the second and following layers are random, the predictive distribution of the full model is non-Gaussian. In fact, this predictive distribution is intractable. Nevertheless, one can easily sample from it by propagating samples through the IP network displayed in Figure 13. This enables a Monte Carlo approximation of the variational objective which can be optimized using stochastic techniques, as in DGPs (Salimbeni & Deisenroth, 2017). Generating the required samples is straightforward given that the variational posterior depends only on the output of the the previous layers. This results in an iterative sampling procedure that can be conducted in an scalable manner. Importantly, the direct evaluation of the covariances are not needed in DVIP, further reducing its cost compared to that of DGPs. The predictive distribution is a mixture of Gaussians (non-Gaussian) and more flexible than that of VIPs. Having a more flexible predictive distribution enables DVIPs to provide more accurate estimations of the uncertainty of the predictions, which is of real use when taking sensible decisions based on the model outcome.

The conducted experiments have shown that DVIPs achieve similar results compared with DGPs at a lower computational cost. Moreover, DVIPs have shown to not over-fit to the observed data by increasing the number of layers and to perform better than single-layer VIPs with a more flexible prior. Moreover, the usage of domain specific priors such as convolutional networks have shown to give remarkable results on image classification datasets compared to other GP methods. This fact sets the usage of not-so-general priors in implicit processes as a new field of study.

Part I

BACKGROUND

In this part, the underlying variational Bayesian theory is reviewed alongside with the approach made in *Gaussian processes* and *variational implicit processes*.

Finally, a thorough review of related research in the topic is conducted, including a detailed discussion of the properties and limitations of the latest state-of-the-art models.

1

BAYESIAN INFERENCE

1.1 INTRODUCTION

Supervised learning problems aim to infer an unknown objective function $f : \mathbb{R}^M \to \mathbb{R}^D$ given a set of paired samples $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ of features $\mathbf{x}_n \in \mathbb{R}^M$ and target values $y_n \approx f(\mathbf{x}_n)$. In order to accomplish this task, it is common to consider a *parametric approach*, where a mapping function $g : \mathbb{R}^M \times \boldsymbol{\Theta} \to \mathbb{R}^D$ that relies on a set of parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ is used to approximate the unknown function f. The objective is to find the optimal values of the parameters $\boldsymbol{\theta}^\star$ that better approximate the desired output, $g(\mathbf{x}, \boldsymbol{\theta}^\star) \approx f(\mathbf{x})$.

In this context, the Bayesian approach considers a set of *unknown or latent variables* $\mathbf{z}$, a prior distribution $P(\mathbf{z})$ and a likelihood distribution $P(y|\mathbf{x}, \mathbf{z})$ that relate the observed variables with the latent ones. Using these distributions and Bayes' rule, a posterior distribution, $P(\mathbf{z}|\mathbf{X}, \mathbf{y})$, can be derived, where $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $\mathbf{y} = \{y_1, \ldots, y_N\}$ are usually assumed to be i.i.d samples. However, in most complex models, the posterior distribution is intractable and must be approximated. An example related with the previous supervised learning setup is to consider $\mathbf{z} = \boldsymbol{\theta}$ and $P(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|g(\mathbf{x}, \boldsymbol{\theta}), \sigma^2)$, making Bayesian inference over the parameters defining the mapping function.

Example 1.1. *An example of mapping function g are artificial neural networks. A neural network (NN) with L hidden layers can be defined as a deterministic non-linear function g parameterized by a set of matrices and vectors $\boldsymbol{\theta} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_L\}$ and non-linear activation functions $\{r_1, \ldots, r_L\}$. Given an input $\mathbf{x}$ the output of the network is calculated as*

$$g(\boldsymbol{\theta}, \mathbf{x}) = h_L \quad \textit{where} \quad h_l = r_l(\boldsymbol{W}_l^T h_{l-1} + \boldsymbol{b}_l) \quad \forall l = 1, \ldots, L\,, \tag{1.1}$$

and $h_0 = \mathbf{x}$. This structure is extended to a Bayesian approach by considering a prior distribution over the parameters $P(\boldsymbol{\theta})$, leading to a Bayesian Neural Network (BNN). An output of the BNN is easily computed by taking a sample from the parameters distribution $\hat{\boldsymbol{\theta}} \sim P(\boldsymbol{\theta})$ and computing the propagation of the input using these parameters, $g(\hat{\boldsymbol{\theta}}, \mathbf{x})$. However, the posterior distribution may not have a closed form in many cases, impeding exact Bayesian inference.

Variational inference (VI, Jordan *et al.* (1999)) exchanges the inference issues related to the Bayesian approach with an optimization problem. In VI, a family of distributions $\mathcal{Q}$ over

the latent variables $\mathbf{z}$ needs to be fixed. The aim is to find the element that minimizes its Kullback-Leibler divergence with the intractable posterior $P(\mathbf{z}|\boldsymbol{X}, \mathbf{y})$ within that family:

$$Q^\star = \arg\min_{Q \in \mathcal{Q}} KL\Big(Q(\mathbf{z}) \mid P(\mathbf{z}|\boldsymbol{X}, \mathbf{y})\Big). \tag{1.2}$$

Definition 1.1 (Kullback-Leibler divergence). Let P and Q be two probability distributions over the same probability space Ω, the Kullback-Leibler divergence $\mathrm{KL}(Q|P)$ measures the *dissimilarity* between both distributions as

$$KL\Big(Q \mid P\Big) = \mathbb{E}_Q\Big[\log Q(x) - \log P(x)\Big]. \tag{1.3}$$

The Kullback-Leibler divergence is defined if and only if for all $x \in \Omega$ such that $P(x) = 0$, then $Q(x) = 0$. In measure terms, Q is absolutely continuous with respect to P.

Proposition 1.1. The Kullback-Leibler divergence is always non-negative.

Proof. As the logarithm is bounded by $x - 1$, we can bound $\log \frac{P(x)}{Q(x)}$

$$\log x \leq x - 1 \implies \frac{P(x)}{Q(x)} - 1 \geq \log \frac{P(x)}{Q(x)}. \tag{1.4}$$

Since probabilities are non-negative, we can multiply by $Q(x)$ in the last inequality

$$P(x) - Q(x) \geq Q(x) \log \frac{P(x)}{Q(x)} = Q(x) \log P(x) - Q(x) \log Q(x). \tag{1.5}$$

Now, integrating both sides

$$0 \geq \mathbb{E}_Q\Big[\log P(x) - \log Q(x)\Big] \implies \mathbb{E}_Q\Big[\log Q(x) - \log P(x)\Big] \geq 0. \tag{1.6}$$

Moreover, the Kullback-Leibler divergence is 0 if and only if the two distributions are equal almost everywhere. □

Variational Bayesian methods or simply *variational methods* are primarily used for two purposes:

1. Perform statistical inference over the unobserved variables by providing an analytical approximation to their posterior probability.
2. Derive and compute a lower bound for the observed data marginal likelihood, that is, the marginal probability of the observed data over the unobserved variables. This is commonly used to perform model selection, where a model with a higher marginal likelihood has a greater probability for being the model that generated the data.

These Q distributions are typically referred as *variational distributions*[1] of the optimization problem. The Kullback-Leibler divergence between the variational distribution $Q(\mathbf{z})$ and the real distribution $P(\mathbf{z}|\mathbf{X}, \mathbf{y})$ may be decomposed in the following way:

$$\begin{aligned} KL\Big(Q(\mathbf{z}) \mid P(\mathbf{z}|\mathbf{X}, \mathbf{y})\Big) &= \mathbb{E}_{Q(\mathbf{z})}\Big[\log Q(\mathbf{z})\Big] - \mathbb{E}_{Q(\mathbf{z})}\Big[\log P(\mathbf{z}|\mathbf{X}, \mathbf{y})\Big] \\ &= \mathbb{E}_{Q(\mathbf{z})}\Big[\log Q(\mathbf{z})\Big] - \mathbb{E}_{Q(\mathbf{z})}\Big[\log P(\mathbf{y}, \mathbf{z}|\mathbf{X}) - \log P(\mathbf{y}|\mathbf{X})\Big] \\ &= \mathbb{E}_{Q(\mathbf{z})}\Big[\log Q(\mathbf{z})\Big] - \mathbb{E}_{Q(\mathbf{z})}\Big[\log P(\mathbf{y}, \mathbf{z}|\mathbf{X})\Big] + \log P(\mathbf{y}|\mathbf{X}). \end{aligned} \tag{1.7}$$

1 The term *variational* comes from *calculus of variations*, a field that studies optimization over functions.

Although the Kullback-Leibler divergence cannot be computed since $P(\mathbf{z}|\mathbf{X}, \mathbf{y})$ is unknown, an equivalent objective can be optimized: its positiveness can be used to set the following lower bound to the evidence, defined as *Evidence Lower Bound* or *ELBO*.

$$\log P(\mathbf{y}|\mathbf{X}) \geq \underbrace{-\mathbb{E}_{Q(\mathbf{z})}\Big[\log Q(\mathbf{z})\Big]}_{Entropy} + \underbrace{\mathbb{E}_{Q(\mathbf{z})}\Big[\log P(\mathbf{y}, \mathbf{z}|\mathbf{X})\Big]}_{Energy} := \text{ELBO}\,.^{2} \tag{1.8}$$

As $P(\mathbf{y}|\mathbf{X})$ does not depend on Q, minimizing the Kullback-Leibler divergence is equivalent to maximize the ELBO, where equality holds if and only if $Q(\mathbf{z}) = P(\mathbf{z}|\mathbf{X}, \mathbf{y})$. The ELBO may be written as

$$\begin{aligned} \text{ELBO}(Q) &= \mathbb{E}_{Q(\mathbf{z})}\Big[\log P(\mathbf{z})\Big] + \mathbb{E}_{Q(\mathbf{z})}\Big[\log P(\mathbf{y}|\mathbf{z}, \mathbf{X})\Big] - \mathbb{E}_{Q(\mathbf{z})}\Big[\log Q(\mathbf{z})\Big] \\ &= \mathbb{E}_{Q(\mathbf{z})}\Big[\log P(\mathbf{y}|\mathbf{z}, \mathbf{X})\Big] - KL\Big(Q(\mathbf{z}) \mid P(\mathbf{z})\Big)\,, \end{aligned} \tag{1.9}$$

where it is expressed as the sum of the log likelihood of the observations and the Kullback-Leibler divergence between the prior $P(\mathbf{z})$ and the variational distribution $Q(\mathbf{z})$. This formula can be further examined: a variational distribution that maximized the first term is the one that makes the observed data most probable; in contrast, the variational distribution that maximizes the second term is $Q(\mathbf{z}) = P(\mathbf{z})$, the prior. This gives the intuitive idea that the KL divergence in the second term works as a regularizer for the variational distribution, i.e, this distributions needs to explain the data (maximize the first term) without diverging from the prior distribution (maximize the second term).

1.2 MEAN FIELD FAMILY

In most cases, the optimization of the ELBO is intractable if a complex variational family distributions is selected. A common assumption is that $\mathcal{Q}$ is the *mean-field variational family*. There, $\mathcal{Q}$ is defined as the family of distributions where the variables are mutually independent, i.e, any $Q \in \mathcal{Q}$ verifies

$$Q(\mathbf{z}) = \prod_{m=1}^{M} Q_m(z_m), \tag{1.10}$$

where $\mathbf{z} = \{z_1, \ldots, z_M\}$ is the considered set of latent variables. Notice that each *factor* Q_m can be different and this family does not depend on the observed data.

The mean-field family can capture any marginal of the latent variables but not correlation between them, as it assumes they are independent. For example, consider a two dimensional Gaussian distribution where most of the density is concentrated inside the blue ellipse shown in Figure 2. A mean-field approximation would factorize as a product of two Gaussian distributions, condensing its density in a circle as shown in purple.

Notice the parametric form of each factor Q_m is not specified and the appropriate configuration depends on the variable. For example, a continuous variable might have a Gaussian factor and a categorical variable have a categorical factor.

2 Energy and Entropy terms come from statistical physics terminology (Barber (2012))

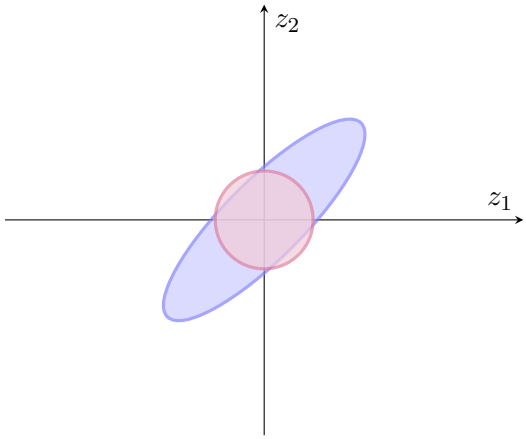


Figure 2: Mean-field family distribution (purple) approximating a Gaussian distribution (blue).

1.3 BLACK BOX α-ENERGY

Variational inference exchanges the inference problem into an optimization problem. To this end, the KL-divergence from the approximate posterior $Q(\mathbf{z})$ to the true posterior $KL(Q(\mathbf{z}) \mid P(\mathbf{z}|\mathbf{X}, \mathbf{y}))$ is minimized. Built upon the idea of VI, BB-α is a modern black-box variational inference framework that generalizes VI. Moreover it unifies and interpolates between VI and expectation propagation (EP) algorithms (Minka, 2013; Li *et al.*, 2015). However, EP is out of the scope of this work. BB-α performs approximate inference by minimizing the following α-divergence (Li & Turner, 2016; Zhu & Rohwer, 1995):

$$D_\alpha\Big(P \mid Q\Big) = \frac{1}{\alpha(1-\alpha)}\left(1 - \int P(\mathbf{z})^\alpha Q(\mathbf{z})^{1-\alpha} d\,\mathbf{z}\right). \tag{1.11}$$

Proposition 1.2. The defined α-divergence generalizes the Kullback-Leibler divergences in the sense that

$$\lim_{\alpha\to 0} D_\alpha\Big(P \mid Q\Big) = KL\Big(Q \mid P\Big) \quad \text{and} \quad \lim_{\alpha\to 1} D_\alpha\Big(P \mid Q\Big) = KL\Big(P \mid Q\Big). \tag{1.12}$$

Proof. Using L'Hôpital's and Leibniz integral rules,

$$\begin{aligned}\lim_{\alpha\to 0} D_\alpha\Big(P \mid Q\Big) &= \lim_{\alpha\to 0} -\frac{1}{1-2\alpha}\int P(\mathbf{z})^\alpha Q(\mathbf{z})^{1-\alpha}\log\frac{P(\mathbf{z})}{Q(\mathbf{z})} d\,\mathbf{z} \\ &= -\int Q(\mathbf{z})\log\frac{P(\mathbf{z})}{Q(\mathbf{z})} d\,\mathbf{z} = KL\Big(Q \mid P\Big).\end{aligned} \tag{1.13}$$

On the other hand, noticing that taking $= D_{1-\alpha}(P \mid Q) = D_\alpha(Q \mid P)$, gives

$$\lim_{\alpha\to 1} D_\alpha\Big(P \mid Q\Big) = \lim_{\alpha\to 0} D_{1-\alpha}\Big(P \mid Q\Big) = \lim_{\alpha\to 0} D_\alpha\Big(Q \mid P\Big) = KL\Big(P \mid Q\Big). \tag{1.14}$$

□

BB-α divergences generalize the divergence measures employed in standard optimization-based approximate inferencem i.e, $KL(P|Q)$, used in EP, and $KL(Q|P)$, used in VI. However, minimizing the α-divergence between a variational distribution and the true posterior is

intractable analitically. As a result, approximated local minimization (factor to factor) is used (Minka, 2005). As shown in Li & Gal (2017), this objective can be approximated using a dataset of size N, as

$$\mathcal{L}_\alpha(Q) = R_\beta\Big(Q(\mathbf{z}) \mid P(\mathbf{z})\Big) - \frac{1}{\alpha}\sum_{n=1}^{N} \log \mathbb{E}_{Q(\mathbf{z})}[P(y_n|\mathbf{x}_n, \mathbf{z})^\alpha]\,, \tag{1.15}$$

where $R_\beta(Q(\mathbf{z})|P(\mathbf{z}))$ denotes the Rényi divergence (Rényi, 1961) of order $\beta = \frac{N}{N-\alpha}$,

$$R_\beta\Big(Q(\mathbf{z}) \mid P(\mathbf{z})\Big) = \frac{1}{\beta - 1} \log \int Q(\mathbf{z})^\beta P(\mathbf{z})^{1-\beta}\, d\mathbf{z}\,. \tag{1.16}$$

This divergence verifies

$$\lim_{\frac{\alpha}{N}\to 0} R_\beta\Big(Q(\mathbf{z}) \mid P(\mathbf{z})\Big) \to KL\Big(Q(\mathbf{z}) \mid P(\mathbf{z})\Big)\,. \tag{1.17}$$

In practice, $\mathcal{L}_\alpha(Q)$ is employed approximating the Rényi divergence term with the standard KL:

$$\mathcal{L}_\alpha(Q) \approx KL\Big(Q(\mathbf{z}) \mid P(\mathbf{z})\Big) - \frac{1}{\alpha}\sum_{n=1}^{N} \log \mathbb{E}_{Q(\mathbf{z})}[P(y_n|\mathbf{x}_n, \mathbf{z})^\alpha]\,. \tag{1.18}$$

These divergences have shown to give better estimations of the log marginal likelihood (Li & Turner, 2016; Santana & Hernández-Lobato, 2022). There is one last consideration regarding this kind of divergences: in many cases the expectations with respect to the variational distribution in Equations (1.9) and (1.18) could not be computable and must be approximated. Under this circumstances, Equation (1.9) is easily approached with Monte-Carlo methods. However, using Monte Carlo on Equation (1.18) leads to biased stochastic gradients because of the non-linear transformation of the logarithm. Nevertheless, this bias can be reduced by increasing the number of Monte-Carlo samples used to approximate the expectation (Hernandez-Lobato *et al.*, 2016).

2

GAUSSIAN PROCESSES

2.1 MOTIVATION AND DEFINITION

Let us first introduce GPs in the context of regression for supervised learning, since they play a key role to understand other key components of this work. An important property of Gaussian processes is that they are equivalent to a Bayesian single-layer NN with infinite width (Neal, 1996). These are, in turn, universal approximators (Hornik *et al.*, 1989), which motivates their usage.

Consider a collection of random variables $(X_1, \ldots, X_n)$ such that their joint distribution is Gaussian,

$$(X_1, \ldots, X_n) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \,. \tag{2.1}$$

The marginalization of this previous Gaussian distribution is also jointly Gaussian,

$$\forall \mathcal{J} \subset \{1, \ldots, n\} \; \exists \boldsymbol{\mu}_{\mathcal{J}}, \boldsymbol{\Sigma}_{\mathcal{J}} \text{ such that } (X_j)_{j \in \mathcal{J}} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathcal{J}}, \boldsymbol{\Sigma}_{\mathcal{J}}) \,. \tag{2.2}$$

This property can be seen the other way around, where the initial collection of variables $(X_1, \ldots, X_n)$ is the marginalization of a bigger (tentatively infinite) collection of variables. From this intuitive idea raises the definition of GPs.

Definition 2.1 (Gaussian process). A GP is a collection of random variables such that any finite subset is jointly Gaussian.

From its definition, Gaussian processes induce a probability distribution over functions, more precisely, over the set of evaluations of functions. Consider a function $f : \mathbb{R}^M \to \mathbb{R}$, such that $f(\cdot) \subset \mathbb{R}$ follows a Gaussian process distribution defined by a mean function m and covariance (or kernel) function κ, i.e, $f \sim \mathcal{GP}(m(\cdot), \kappa(\cdot, \cdot))$. Following the definition, if $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T \subset \mathbb{R}$ is a finite subset of $\mathbb{R}$, then $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^T$ follows a Gaussian distribution,

$$\mathbf{f} \sim \mathcal{N}(m(\mathbf{X}), \kappa(\mathbf{X}, \mathbf{X})). \tag{2.3}$$

Learning finite m-dimensional Gaussians require $m + m(m+1)/2$ parameters (m for the mean value and $m(m+1)/2$ for the symmetric covariance matrix). For this reason, learning with parametric models is usually an *overdetermined* problem (e.g. linear regression). In contrast, under the usual regression supervised learning paradigm, the usage of Gaussian processes lies on the assumption that the unknown function f is an observation (or realization) of a Gaussian process of unknown mean and kernel functions (Figure 3) . As a result, the

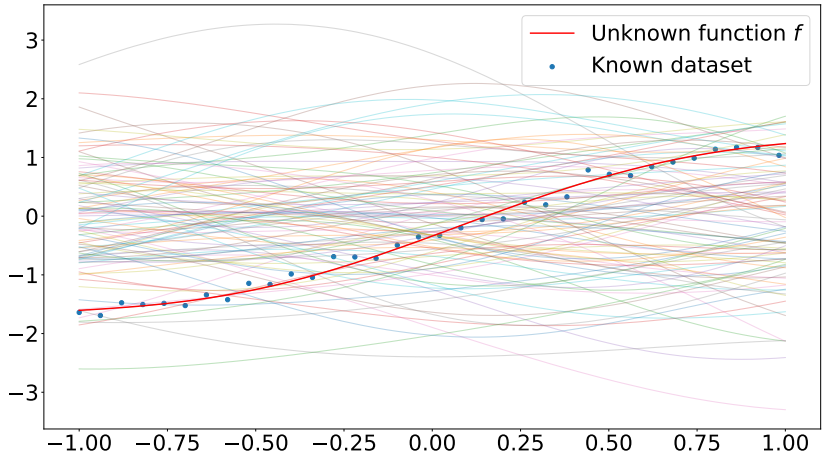


Figure 3: The usual learning hypothesis for GPs is that the unknown function f is a sample from an unknown GP distribution and the given dataset is a noisy subset of the evaluation of this function.

functional approach based on Gaussian processes aims to learn an infinite-dimensional distribution from not even a single sample (which would correspond to a full function, not a finite set of samples from it). Therefore, learning a GP is an *underdetermined* problem.

In order to bypass this difficulty, it is common practise to determine a parametric family for the mean and kernel functions. More precisely, the mean function is usually zero without loss of generality, and the kernel function is usually in a parametric family of kernels, e.g., the radial basis function (RBF) kernel (Example shown in Figure 4). This kernel is characterized for its smoothness, which is a consequence of it being infinitely derivable. More precisely, the closed form of the RBF kernel only depends on two parameters l and σ, and is defined as,

$$\mathrm{RBF}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2l^2}\right). \tag{2.4}$$

The impact of these parameters in the kernel are shown in Figure 5. In these figures, the length-scale l is seen to affect the correlation of points with further points, making the functions smoother as this parameter increases. On the other hand, the variance σ affects the intensity or amplitude of the kernel.

2.2 LEARNING A GAUSSIAN PROCESS

In practise, modelling data as a Gaussian process with a smooth kernel may result in miss-modelling (Williams & Rasmussen, 2006). Figure 6 shows the predictive mean of two GPs with RBF kernel with and without noise consideration. To account for model mismatch, a Gaussian likelihood is commonly used for regression; defining a GP prior over the function $f(\cdot) \sim \mathcal{GP}(m(\cdot), \kappa(\cdot,\cdot))$, an observation y is modelled as

$$y = f(\mathbf{x}) + \epsilon, \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{2.5}$$

Assume a training dataset $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \mathbf{y} = \{y_1, \ldots, y_N\}$ of inputs and noisy outputs of an unknown function f is given. Prediction using new input values $\mathbf{x}^\star$ is straightforward:

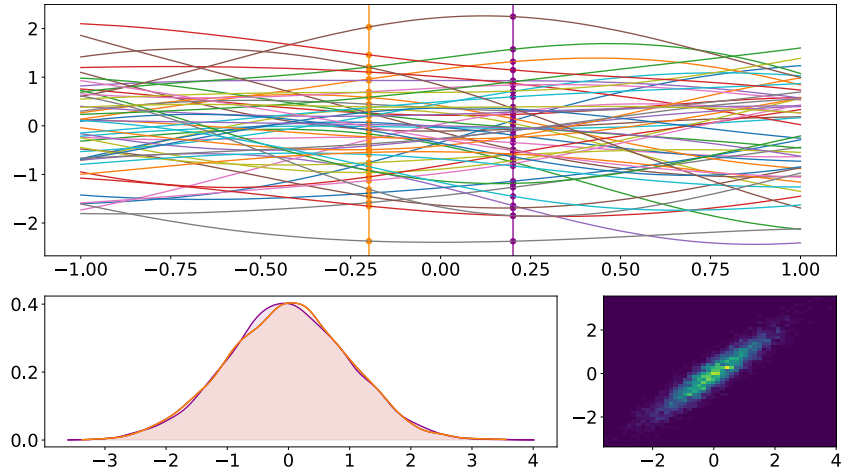


Figure 4: Samples from a GP with zero mean and RBF kernel (top). Empirical uni-dimensional distributions obtained at the marked points (bottom left) and two-dimensional (bottom right).

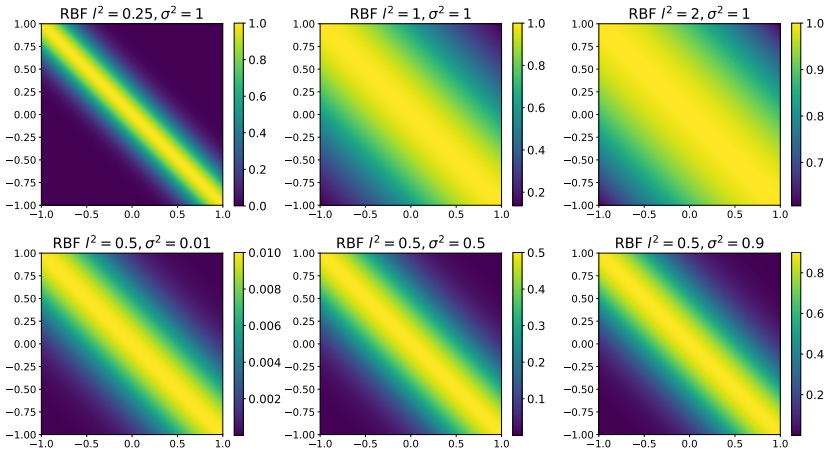


Figure 5: Radial Basis Exponential (RBF) kernel in a grid $[-1, 1] \times [-1, 1]$ for different hyperparameter values.

the set of known training values $\mathbf{f} = f(\mathbf{x})$ and the unknown test values $\mathbf{f}^* = f(\mathbf{x}^*)$ are finite subsets of the Gaussian Process and verify

$$(\mathbf{f}, \mathbf{f}^*) \sim \mathcal{N}\left(\begin{pmatrix} m(\mathbf{x}) \\ m(\mathbf{x}^\star) \end{pmatrix}, \begin{pmatrix} \kappa(\mathbf{X}, \mathbf{X}) & \kappa(\mathbf{X}, \mathbf{x}^\star) \\ \kappa(\mathbf{x}^\star, \mathbf{X}) & \kappa(\mathbf{x}^\star, \mathbf{x}^\star) \end{pmatrix}\right). \tag{2.6}$$

Moreover, the predictive distribution can be computed in closed form. More precisely, the joint distribution of the data and the prediction $P(\mathbf{y}, \mathbf{f}^\star)$ is Gaussian (Murphy, 2012)

$$\begin{aligned} P(\mathbf{y}, \mathbf{f}^\star) &= \int_{\mathbf{f}} P(\mathbf{y}|\mathbf{f}) P(\mathbf{f}, \mathbf{f}^\star) \\ &= \mathcal{N}\left(\begin{pmatrix} m(\mathbf{x}) \\ m(\mathbf{x}^\star) \end{pmatrix}, \begin{pmatrix} \kappa(\mathbf{X}, \mathbf{X}) + \sigma^2 \boldsymbol{I} & \kappa(\mathbf{X}, \mathbf{x}^\star) \\ \kappa(\mathbf{x}^\star, \mathbf{X}) & \kappa(\mathbf{x}^\star, \mathbf{x}^\star) \end{pmatrix}\right), \end{aligned} \tag{2.7}$$

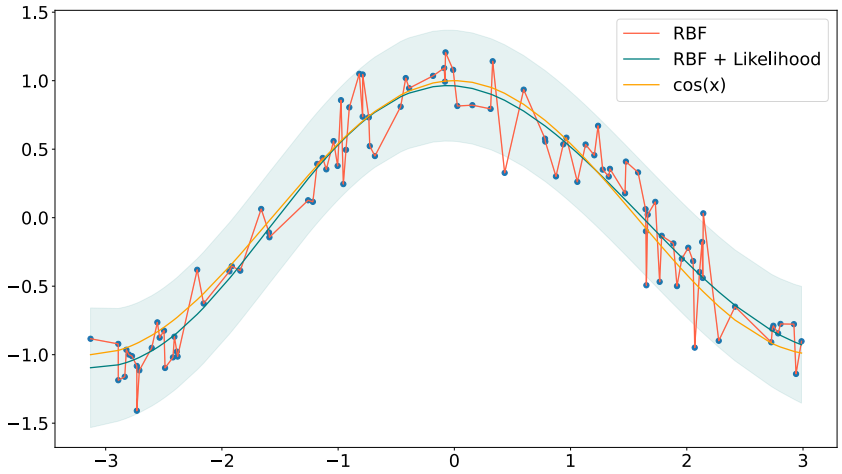


Figure 6: Gaussian process with RBF kernel, with and without Gaussian likelihood, learning a noisy cosine function. The uncertainty learned by the GP that considers a likelihood in shown in gray. The GP without likelihood has a negligible uncertainty estimation.

where the last equality can be shown using the fact that the convolution of Gaussian distributions is Gaussian. Using that the conditional of a multivariate Gaussian distribution is Gaussian, leads to the following predictive distribution

$$P(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}^\star, \boldsymbol{\Sigma}^\star)\,, \tag{2.8}$$

with mean and covariance matrix given by

$$\begin{aligned} \boldsymbol{\mu}^\star &= m(\mathbf{x}^\star) + \kappa(\mathbf{x}^\star, \mathbf{X})\big(\kappa(\mathbf{X}, \mathbf{X}) + \sigma^2 \boldsymbol{I}\big)^{-1}(\mathbf{y} - m(\mathbf{X}))\,, \\ \boldsymbol{\Sigma}^\star &= \kappa(\mathbf{x}^\star, \mathbf{x}^\star) - \kappa(\mathbf{x}^\star, \mathbf{X})\big(\kappa(\mathbf{X}, \mathbf{X}) + \sigma^2 \boldsymbol{I}\big)^{-1}\kappa(\mathbf{X}, \mathbf{x}^\star)\,. \end{aligned} \tag{2.9}$$

Figure 7 shows an example of prior and predictive distribution using zero mean and RBF kernel.

2.3 HYPER-PARAMETERS LEARNING

Up to this point, the predictive distribution of GPs has being studied. However, how can the hyper-parameters of the likelihood as well as the mean and kernel functions be tuned? To answer this question, consider a suitable likelihood (Gaussian for regression) and a mean and kernel function for the GP. Using this, it is easy to derive an exact expression for the marginal likelihood:

$$\log P(\mathbf{y}) = \int P(\mathbf{y}|\mathbf{f}) P(\mathbf{f})\, d\mathbf{f}\,, \tag{2.10}$$

where $P(\mathbf{f})$ is a Gaussian Process prior with $\kappa(\mathbf{X}, \mathbf{X}) = \boldsymbol{K}$ and $P(\mathbf{y}|\mathbf{f})$ is a Gaussian likelihood defined by some noise variance σ^2:

$$P(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \boldsymbol{K})\,, \quad P(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma^2 \boldsymbol{I})\,. \tag{2.11}$$

As a result, the marginal likelihood follows a Gaussian distribution

$$P(\mathbf{y}) = \mathcal{N}(\mathbf{0}, \boldsymbol{K} + \sigma^2 \boldsymbol{I})\,, \tag{2.12}$$

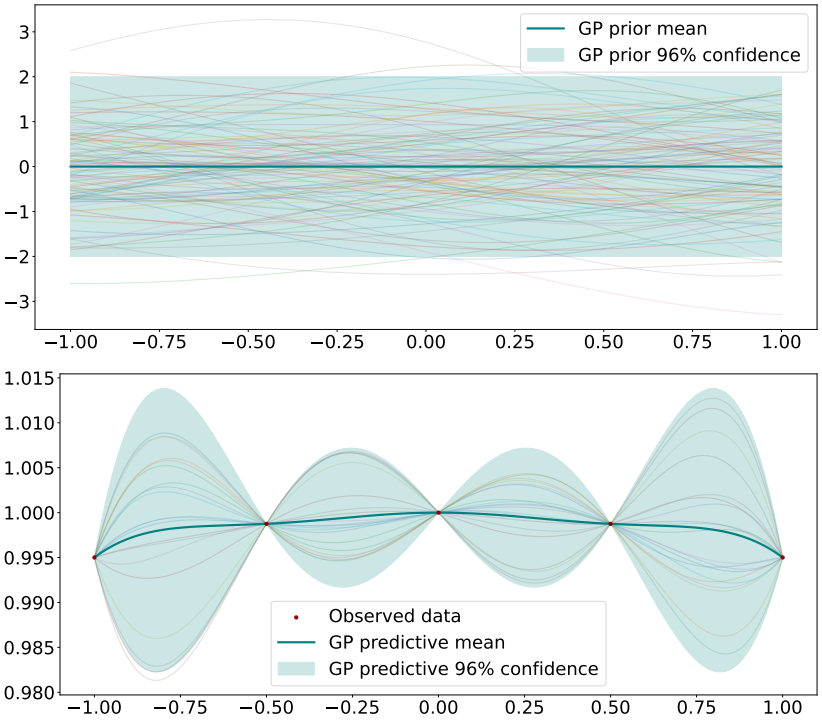


Figure 7: Gaussian process prior distribution (above) and predictive distribution with five observations (below) using a RBF kernel and zero mean.

whose logarithm has closed form:

$$\log P(\mathbf{y}) = -\frac{1}{2}\mathbf{y}^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\mathbf{y} - \frac{1}{2}\log\det(\boldsymbol{K} + \sigma^2\boldsymbol{I}) - \frac{N}{2}\log 2\pi\,, \tag{2.13}$$

and can be optimized using gradient methods. The three terms in Equation (2.13) can be easily interpreted; the first term is the only one considering the observed target values and computes how much the model explains the data, the second term is a *complexity penalty* term, that depends only on the covariance function. The last term is just a normalization constant (Rasmussen & Williams, 2006).

Kernel parameters can be also tuned using the precise kernel expression in Equation (2.13). For example, if an RBF kernel is used, both parameters l and σ can be set to better explain the data. For example, in terms of Equation (2.13), using a bigger length-scale makes less flexible models, decreasing the data-fitting term and the complexity penalty term (Rasmussen, 2003). On the other hand, using a lower length-scale makes the model more flexible, increasing the complexity penalty and the data-fitting term (Figure 8).

2.4 COMPUTATIONAL EFFICIENCY

The main drawback of using the exact predictive distribution is its computational complexity. As kernel method (Shawe-Taylor *et al.*, 2004), training requires the computation of the inverse of a $N \times N$ matrix, which require $\mathcal{O}(N^3)$ operations. On the other hand, prediction requires $\mathcal{O}(N^2)$ operations per test point. As a result, using exact inference in Gaussian Processes is impracticable on medium to large datasets.

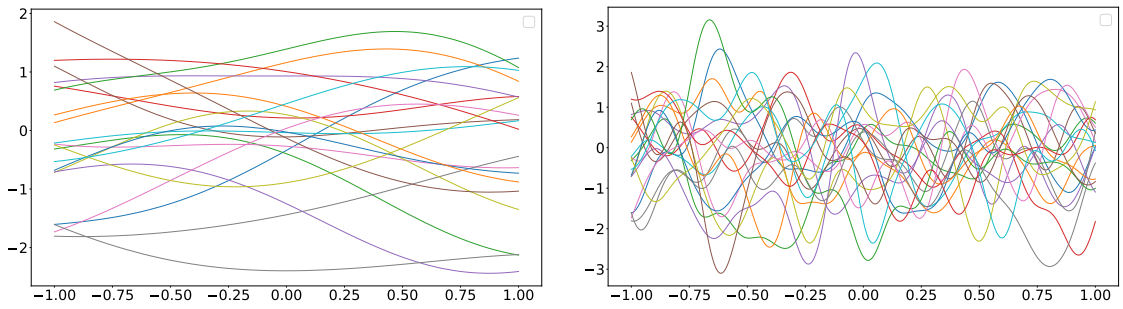


Figure 8: GP samples with RBF kernel and length-scale 1 (left) and 0.1 (right). This shows how lower length-scales induce more flexible functions.

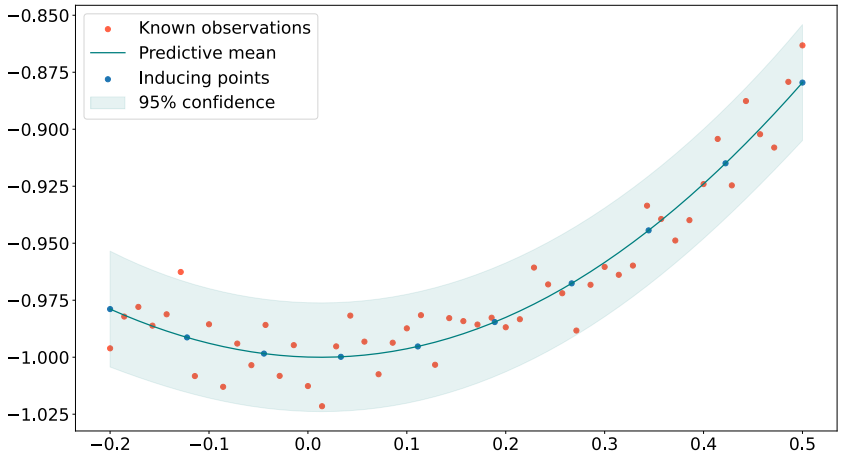


Figure 9: Gaussian process with inducing points example. The location of these inducing points as being set to be equally spaced.

In the following sections two of the most common approaches to improve the basic scalability of GPs are reviewed. These are:

1. The usage of sparse inducing inputs and variational inference (Snelson & Ghahramani, 2006; Titsias, 2009; Hensman *et al.*, 2013; Bui *et al.*, 2016), which reduce the training efficiency to be cubically on the number of inducing points.
2. The approximation of the log marginal likelihood of the GP, $P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$, by the log marginal likelihood of a Bayesian linear regression model with S randomly sampled prior values (Ma *et al.*, 2019). This approach based on prior samples will be further used with implicit processes.

2.5 INDUCING POINTS APPROXIMATION

The most common approach to overcome the computational complexity of GPs are the so called *sparse Gaussian processes* (SGPs) (Snelson & Ghahramani, 2006), which are later used in the experiments of this work. These models assume a set of *inducing variables* $\mathbf{Z} \subset \mathbb{R}^M$ in the same space of the training data features $\mathbf{X}$. These inducing variables are often initialized using any cluster algorithm, for example, K-Means, with $|\mathbf{Z}| << |\mathbf{X}|$. The point of this approximation is that learning is cubic with respect to $|\mathbf{Z}|$ rather than $|\mathbf{X}|$ (Figure 9).

In general, SGPs consider this set of pseudo-inputs and their corresponding unknown function values $\mathbf{u} = \mathbf{f}(\mathbf{Z})$. Inference is made marginalizing these function values. In this document we are reviewing the variational inference approximation using inducing points, that is, approximated inference over the exact model. However, there are methods of performing exact inference over an approximated model (Quiñonero-Candela & Rasmussen, 2005).

Considering a variational distribution $Q(\mathbf{u}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{A})$ that approximates $P(\mathbf{u}|\mathbf{y})$, leads to the following variational posterior distribution over the latent function values $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^T$,

$$Q(\mathbf{f}) = \int P(\mathbf{f}|\mathbf{u})Q(\mathbf{u})\, d\mathbf{u} \approx \int P(\mathbf{f}|\mathbf{u})P(\mathbf{u}|\mathbf{y})\, d\mathbf{u} = P(\mathbf{f}|\mathbf{y})\,. \tag{2.14}$$

Moreover, the variational approach assumes that $\mathbf{u}$ is a sufficient statistic for the set of testing function values $\mathbf{f}^\star$, i.e, $P(\mathbf{f}^\star|\mathbf{f}, \mathbf{u}) = P(\mathbf{f}^\star|\mathbf{u})$. Leading to the following variational predictive distribution,

$$\begin{aligned} Q(\mathbf{f}^\star) &= \int\int P(\mathbf{f}^\star|\mathbf{u})P(\mathbf{f}|\mathbf{u})Q(\mathbf{u})\, d\mathbf{f}\, d\mathbf{u} = \int P(\mathbf{f}^\star|\mathbf{u})Q(\mathbf{u})\, d\mathbf{u} \\ &= \mathcal{N}(\boldsymbol{\mu}^\star, \boldsymbol{\Sigma}^\star)\,, \end{aligned} \tag{2.15}$$

with mean and covariance matrix given by

$$\begin{aligned} \boldsymbol{\mu}^\star &= \kappa(\mathbf{x}^\star, \mathbf{Z})\kappa(\mathbf{Z}, \mathbf{Z})^{-1}\boldsymbol{\mu}\,, \\ \boldsymbol{\Sigma}^\star &= \kappa(\mathbf{x}^\star, \mathbf{x}^\star) - \kappa(\mathbf{x}^\star, \mathbf{Z})\kappa(\mathbf{Z}, \mathbf{Z})^{-1}\kappa(\mathbf{Z}, \mathbf{x}^\star) \\ &\quad + \kappa(\mathbf{x}^\star, \mathbf{Z})\kappa(\mathbf{Z}, \mathbf{Z})^{-1}\mathbf{A}\kappa(\mathbf{Z}, \mathbf{Z})^{-1}\kappa(\mathbf{Z}, \mathbf{x}^\star)\,. \end{aligned} \tag{2.16}$$

Variational optimization is therefore performed by optimizing the ELBO, which simplifies to the following expression (Salimbeni & Deisenroth, 2017):

$$\mathcal{L} = \mathbb{E}_{Q(\mathbf{f})}\Big[\log P(\mathbf{y}|\mathbf{f})\Big] - KL\Big(Q(\mathbf{u}) \mid P(\mathbf{u})\Big)\,. \tag{2.17}$$

Where $P(\mathbf{u})$ is usually set to a standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. This expression can be evaluated with cost $\mathcal{O}(N|\mathbf{Z}|^2)$ instead of $\mathcal{O}(N^3)$.

2.6 BAYESIAN LINEAR REGRESSION APPROXIMATION

Another popular approximation to improve GP's scalability is the usage of a Bayesian linear regression model to approximate the posterior distribution of the Gaussian process (Ma *et al.*, 2019). This approach is based on random feature approximation techniques for Gaussian processes (Balog *et al.*, 2016; Gal & Turner, 2015). This is also similar to *relevance vector machines* in Quiñonero-Candela & Rasmussen (2005), which are equivalent to a degenerative Gaussian Process. The following theorem is needed to understand the approach.

Theorem 2.1 (Mercer's Theorem Mercer (1909)). Suppose K is a continuous symmetric positive-definite kernel on $\mathbb{R}^M$. If the function $\kappa(\mathbf{x}) = K(\mathbf{x}, \mathbf{x})$ is $L^1(\mathbb{R}^M)$, then, there exists an orthogonal set $\{\psi_i\}_{i\in\mathbb{N}}$ of $L^2(\mathbb{R}^M)$, and scalar non-negative values $\{\lambda\}_{i\in\mathbb{N}}$, such that

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=1}^{\infty} \lambda_i \psi_i(\mathbf{x}_1)\psi_i(\mathbf{x}_2)\,. \tag{2.18}$$

From Mercer's theorem, naming $\boldsymbol{\phi}(\mathbf{x}) = (\lambda_i^{1/2}\psi_i(\mathbf{x}))_{i\in\mathbb{N}}$, the kernel can be seeing as the inner product of a set of basis functions

$$\kappa(\mathbf{x},\mathbf{y}) = \boldsymbol{\phi}(\mathbf{x})^T\boldsymbol{\phi}(\mathbf{y})\,. \tag{2.19}$$

Using this expression for the kernel, the GP Gaussian predictive distribution (Equation 2.9) has mean and covariance matrix defined by

$$\begin{aligned}\boldsymbol{\mu}^\star &= m(\mathbf{x}^\star) + \boldsymbol{\phi}(\mathbf{x}^\star)^T\boldsymbol{\phi}(\mathbf{X})\big(\boldsymbol{\phi}(\mathbf{X})^T\boldsymbol{\phi}(\mathbf{X})\big)^{-1}(\mathbf{y} - m(\mathbf{X}))\,,\\ \boldsymbol{\Sigma}^\star &= \boldsymbol{\phi}(\mathbf{x}^\star)^T\boldsymbol{\phi}(\mathbf{x}^\star) - \boldsymbol{\phi}(\mathbf{x}^\star)^T\boldsymbol{\phi}(\mathbf{X})\big(\boldsymbol{\phi}(\mathbf{X})^T\boldsymbol{\phi}(\mathbf{X})\big)^{-1}\boldsymbol{\phi}(\mathbf{X})^T\boldsymbol{\phi}(\mathbf{x}^\star)\,,\end{aligned} \tag{2.20}$$

where $\boldsymbol{\phi}(\mathbf{X}) = (\boldsymbol{\phi}(\mathbf{x}_1),\dots,\boldsymbol{\phi}(\mathbf{x}_N))^T$.

In practise, using the basis functions is an intractable approach. However, it induces a variational approximation for cases when the kernel lacks a closed form but samples from the GP can be obtained. More precisely, consider $f_1,\dots,f_S$ samples from a GP of unknown mean m and kernel function κ. It is clear that if S is large,

$$\begin{aligned} m(\mathbf{x}) &\approx \hat{m}(\mathbf{x}) = \frac{1}{S}\sum_{s=1}^{S} f_s(\mathbf{x})\,,\\ \kappa(\mathbf{x}_1,\mathbf{x}_2) &\approx \frac{1}{S}\sum_{s=1}^{S}\Big((f_s(\mathbf{x}_1) - \hat{m}(\mathbf{x}_1))(f_s(\mathbf{x}_2) - \hat{m}(\mathbf{x}_2))\Big)\,.\end{aligned} \tag{2.21}$$

That is, defining

$$\hat{\boldsymbol{\phi}}(\mathbf{x}) = \frac{1}{\sqrt{S}}\Big(f_1(\mathbf{x}) - \hat{m}(\mathbf{x}),\dots,f_S(\mathbf{x}) - \hat{m}(\mathbf{x})\Big)^T\,, \tag{2.22}$$

the kernel function κ can be approximated as the inner product of "empirical basis functions"

$$\kappa(\mathbf{x}_1,\mathbf{x}_2) \approx \hat{\boldsymbol{\phi}}(\mathbf{x}_1)^T\hat{\boldsymbol{\phi}}(\mathbf{x}_2)\,. \tag{2.23}$$

With this approximation in mind, consider the following Bayesian linear regression model:

$$f(\mathbf{x}) = \hat{m}(\mathbf{x}) + \hat{\boldsymbol{\phi}}(\mathbf{x})^T\boldsymbol{a}, \tag{2.24}$$

where $\boldsymbol{a}$ is a new set of parameters. Let $P(\boldsymbol{a}) = \mathcal{N}(\mathbf{0},\boldsymbol{I})$ be a Gaussian prior distribution and $Q(\boldsymbol{a}) = \mathcal{N}(\boldsymbol{\mu_a},\boldsymbol{S_a})$ a Gaussian variational posterior. As the prior is Gaussian, this induces a GP prior over f with mean and covariance functions given by Equation (2.24). The predictive distribution over a new point $\mathbf{x}^\star$ is (Murphy, 2012),

$$\begin{aligned} Q(\mathbf{f}^\star|\mathbf{x}^\star) &= \int P(\mathbf{f}^\star|\mathbf{x}^\star,\boldsymbol{a})Q(\boldsymbol{a})\,d\boldsymbol{a}\\ &= \mathcal{N}\left(\hat{m}(\mathbf{x}^\star) + \hat{\boldsymbol{\phi}}(\mathbf{x}^\star)^T\boldsymbol{\mu_a}, \hat{\boldsymbol{\phi}}(\mathbf{x}^\star)^T\boldsymbol{S_a}\hat{\phi}(\mathbf{x}^\star)\right).\end{aligned} \tag{2.25}$$

This predictive distribution is similar to the one obtained from the Gaussian process when considering as kernel function the inner product of the basis functions (Equation 2.20). Here, the terms concerning the training dataset $\mathbf{X},\mathbf{y}$ are encapsulated in the variational parameters $\boldsymbol{\mu_a}$ and $\boldsymbol{S_a}$.

Moreover, the optimal values of $\boldsymbol{\mu}_{\mathbf{a}}$ and $\boldsymbol{S}_{\mathbf{a}}$ can be analytically computed in closed form (Ma *et al.*, 2019). In fact, the optimal distribution is

$$\begin{aligned} \boldsymbol{\mu}_{\mathbf{a}}^{\star} &= \frac{1}{\sigma^2} \boldsymbol{S}_{\mathbf{a}}^{\star} \hat{\boldsymbol{\phi}}(\mathbf{X})^T (\mathbf{y} - \hat{\boldsymbol{m}}(\mathbf{X})) \,, \\ \boldsymbol{S}_{\mathbf{a}}^{\star} &= \sigma^2 (\hat{\boldsymbol{\phi}}(\mathbf{X})^T \hat{\boldsymbol{\phi}}(\mathbf{X}) + \sigma^2 \boldsymbol{I})^{-1} \,. \end{aligned} \tag{2.26}$$

However, this analytic solution requires the evaluation of the approximated kernel on the entire dataset, $\hat{\boldsymbol{\phi}}(\mathbf{X})^T \hat{\boldsymbol{\phi}}(\mathbf{X})$, which makes it impossible to compute in large datasets. Furthermore, this analytic evaluation has a computational cost of $\mathcal{O}(NS^2 + S^3)$, given that it requires the inversion of a $S \times S$ matrix.

This difficulty can be bypassed performing variational inference over the linear regression coefficients $\boldsymbol{a}$ in order to learn $\boldsymbol{\mu}_{\mathbf{a}}^{\star}$ and $\boldsymbol{S}_{\mathbf{a}}^{\star}$ and approximate their true values. This approach leads to the optimization of the following ELBO (Ma *et al.*, 2019):

$$\mathcal{L} = \mathbb{E}_{Q(\mathbf{f})} \Big[\log P(\mathbf{y}|\mathbf{f}) \Big] - KL\Big(Q(\mathbf{a}) \;\big|\; P(\mathbf{a}) \Big) \,, \tag{2.27}$$

which is similar to the one computed using inducing points, and differs from it in the evaluation of the first term. Using this approach, the computational cost of each training iteration is $\mathcal{O}(NS^2)$. Moreover, the addition in the first term can be approximated using mini-batches B. This, combined with stochastic optimization has a cost of $\mathcal{O}(|B|S^2)$.

This approach based on prior samples is specially useful in cases where point-wise density of the prior cannot be computed but samples can be easily drawn. This is the case of implicit processes, which are reviewed in the following chapter.

3

VARIATIONAL IMPLICIT PROCESSES

3.1 MODEL DEFINITION

One of the drawbacks of the standard Gaussian process approaches is that the predictive distribution is highly restricted by the selected parametric kernel family (Figure 10). To overcome this limitation, implicit processes define a stochastic process from a highly flexible family of priors, whose main drawback is that they lack a closed expression.

In this section implicit process are introduced as a generalization of the Gaussian process to intractable prior distributions. Gaussian processes can be parameterized using the re-parameterization trick of the Gaussian distribution. That is, if $\mathbf{f}$ is Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, sampling from $\mathbf{f}$ is equivalent to sample from $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and apply the linear transformation $\mathbf{f} = \boldsymbol{B}^T\mathbf{z} + \boldsymbol{\mu}$, where $\boldsymbol{B}$ is the Cholesky decomposition of $\boldsymbol{\Sigma}$. Using this idea, it is intuitive to generalize GPs to a more general definition; where noise is generated from an arbitrary distribution $\mathbf{z} \sim P_{\mathbf{z}}$ that is not fixed to be Gaussian and $\mathbf{f}$ comes from a parametric transformation, that is not necessarily linear.

Definition 3.1 (Implicit Process Ma *et al.* (2019)). An implicit stochastic process (IP) is a collection of random variables $f(\cdot)$ such that any finite collection $\mathbf{f} = \{f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots, f(\mathbf{x}_N)\}$ is implicitly defined by the following generative process:

$$\mathbf{z} \sim P_{\mathbf{z}}(\mathbf{z}) \quad \textit{and} \quad f(\mathbf{x}_n) = g_{\boldsymbol{\theta}}(\mathbf{x}_n, \mathbf{z}),\ \forall n = 1, \ldots, N. \tag{3.1}$$

The above IP is denoted as $f(\cdot) \sim \mathcal{IP}(g_{\boldsymbol{\theta}}(\cdot, \cdot), P_{\mathbf{z}})$, with $\boldsymbol{\theta}$ the IP's parameters, $P_{\mathbf{z}}$ a source of noise, and $g_{\boldsymbol{\theta}}(\mathbf{x}_n, \mathbf{z})$ a function that given $\mathbf{z}$ and $\mathbf{x}_n$ outputs $f(\mathbf{x}_n)$. $g_{\boldsymbol{\theta}}(\mathbf{x}_n, \mathbf{z})$ can be, *e.g.*,

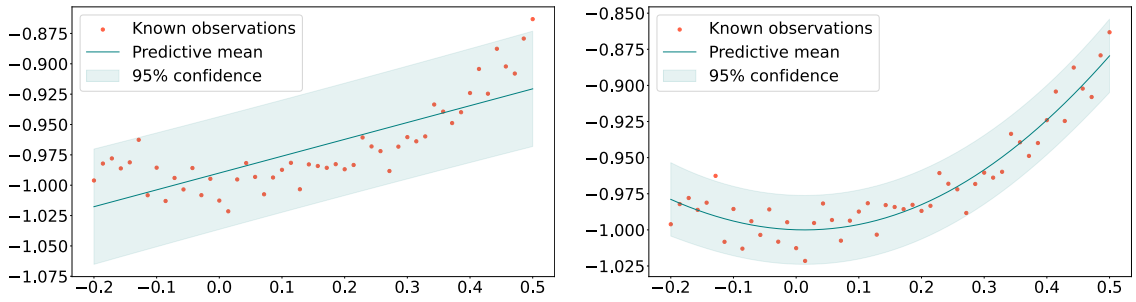


Figure 10: Gaussian process predictive distribution using a linear kernel (left) and an exponential kernel (right).

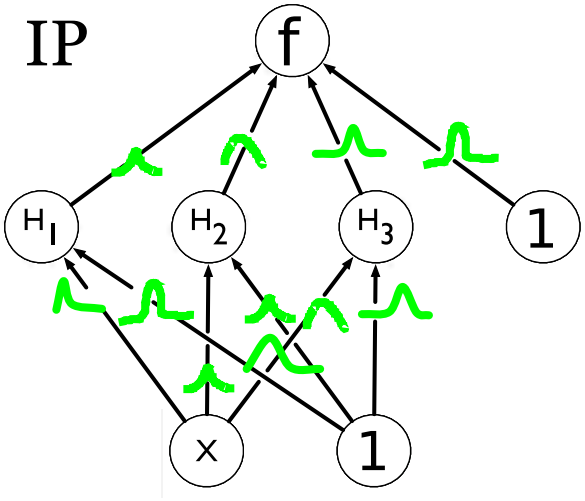


Figure 11: IP resulting from a BNN with random weights and biases following a Gaussian distribution. A sample of the weights and biases generates a random function.

a NN whose weights are computed in terms of $\mathbf{z}$ and $\boldsymbol{\theta}$ using the reparametrization trick (Kingma & Welling, 2014) (Figure 11). In this figure, it is shown a representation of a BNN where biases are encapsulated inside the weights for simplicity. A sample from the BNN can be drawn by sampling from the weights distributions and computing the output using these samples and the input value.

Defining an implicit process distribution over an unknown function $f(\cdot) \sim \text{IP}(g_{\boldsymbol{\theta}}(\cdot,\cdot), P_{\mathbf{z}})$ and using a suitable likelihood $P(\mathbf{y}|\mathbf{f})$ for the problem produces an implicit model $P(\mathbf{y},\mathbf{f}|\mathbf{X})$. This means that there is not an explicit density for that joint distribution, but samples can be easily drawn from it thanks to the IP formulation. Given a sample $\mathbf{z} \sim P_{\mathbf{z}}(\mathbf{z})$ and an input $\mathbf{x}$, it is straightforward to generate a sample $f(\mathbf{x})$ using $g_{\boldsymbol{\theta}}$, *i.e.*, $f(\mathbf{x}) = g_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})$. In this context, both the prior $P(\mathbf{f}|\mathbf{X})$ and the posterior $P(\mathbf{f}|\mathbf{X},\mathbf{y})$ are generally intractable, since the IP assumption does not allow for point-wise density estimation, except cases as that of a GP. To overcome this difficulty, in Ma *et al.* (2019) the model's joint distribution, $P(\mathbf{y},\mathbf{f}|\mathbf{X})$, is approximated by the following distribution:

$$P(\mathbf{y},\mathbf{f}|\mathbf{X}) \approx Q(\mathbf{y},\mathbf{f}|\mathbf{X}) = P(\mathbf{y}|\mathbf{f})Q_{\mathcal{GP}}(\mathbf{f}|\mathbf{X}), \tag{3.2}$$

where $Q_{\mathcal{GP}}$ is a Gaussian process with mean and covariance functions $m(\cdot)$ and $\mathcal{K}(\cdot,\cdot)$, respectively. These two functions are in turn defined by the mean and covariance functions of the IP,

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \quad \mathcal{K}(\mathbf{x}_1,\mathbf{x}_2) = \mathbb{E}\left[\big(f(\mathbf{x}_1) - m(\mathbf{x}_1)\big)\big(f(\mathbf{x}_2) - m(\mathbf{x}_2)\big)\right], \tag{3.3}$$

which can be estimated empirically by sampling from $\text{IP}(g_{\boldsymbol{\theta}}(\cdot,\cdot), P_{\mathbf{z}})$ (Ma *et al.*, 2019). Specifically, using S Monte Carlo samples $f_s(\cdot) \sim \text{IP}(g_{\boldsymbol{\theta}}(\cdot,\cdot), P_{\mathbf{z}})$, $s \in 1,\ldots,S$. The mean and covariance functions are

$$\begin{aligned} m^\star(\mathbf{x}) &= \frac{1}{S}\sum_{s=1}^{S} f_s(\mathbf{x}), \qquad \mathcal{K}^\star(\mathbf{x}_1,\mathbf{x}_2) = \boldsymbol{\phi}(\mathbf{x}_1)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_2), \\ \boldsymbol{\phi}(\mathbf{x}_n) &= \frac{1}{\sqrt{S}}\Big(f_1(\mathbf{x}_n) - m^\star(\mathbf{x}_n), \ldots, f_S(\mathbf{x}_n) - m^\star(\mathbf{x}_n)\Big)^{\mathrm{T}}. \end{aligned} \tag{3.4}$$

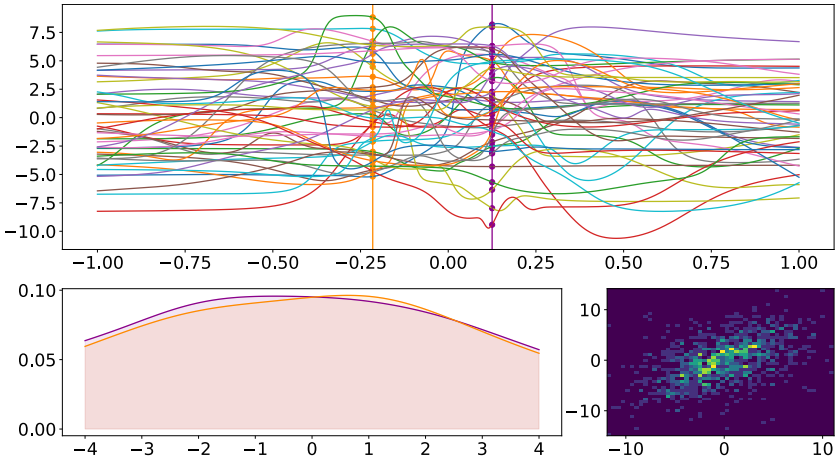


Figure 12: Samples from an implicit process defined by a 2-layer BNN with width 10 and weights and biases following a Gaussian distribution (top). Empirical uni-dimensional distributions obtained at the marked points (bottom left) and two-dimensional (bottom right).

Thus, the prior for f is simply a GP approximating the prior IP, which can be, *e.g.*, a BNN (Figure 12).

Critically, the samples $f_s(\mathbf{x})$ keep the dependence w.r.t. the IP prior parameters $\boldsymbol{\theta}$, which enables prior adaptation to the observed data in VIPs (Ma *et al.*, 2019). Unfortunately, this formulation has the typical cubic cost in the training set size of GPs (Rasmussen & Williams, 2006). To solve this and also allow for mini-batch based optimization, the GP is further approximated using a linear model defined as $f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{a} + m^{\star}(\mathbf{x})$, where $P(\mathbf{a}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ (see Section 2.6 for more details). Under this definition, the prior mean and covariances of $f(\mathbf{x})$ are given by Equation (3.4). The posterior of $\mathbf{a}$, $P(\mathbf{a}|\mathbf{y})$, is approximated using a Gaussian distribution, $Q_{\boldsymbol{\omega}}(\mathbf{a}) = \mathcal{N}(\mathbf{a}|\boldsymbol{m}, \boldsymbol{S})$, whose parameters $\boldsymbol{\omega} = \{\boldsymbol{m}, \boldsymbol{S}\}$ (and other model's parameters) are adjusted by maximizing the α-energy (Section 1.3)

$$\mathcal{L}^{\alpha}(\boldsymbol{\omega}, \boldsymbol{\theta}, \sigma^2) = \sum_{n=1}^{N} \frac{1}{\alpha} \log \mathbb{E}_{Q_{\omega}}\Big[P(y_n|f(\mathbf{x}_n))^{\alpha}\Big] - \mathrm{KL}\Big(Q_{\omega}(\mathbf{a}) \mid P(\mathbf{a})\Big)\,, \tag{3.5}$$

where $\alpha = 0.5$, since such a value provides good general results Hernandez-Lobato *et al.* (2016); Santana & Hernández-Lobato (2022). The Kullback-Leibler divergence in the objective function is easily computable given that both distribution are Gaussian (Section 3.4). Moreover, the uni-variate expectations in the first term are computable analytically in some specific cases, e.g. Gaussian or Poisson likelihoods $P(y_n|f(\mathbf{x}_n))$ (Hensman *et al.*, 2015) (Section 3.5). In other cases, Monte Carlo sampling can be used in order to approximate this expectation. For instance, Bonilla *et al.* (2018) shows this is possible when $Q_{\omega}(\boldsymbol{a})$ is a mixture of Gaussian distributions. Importantly, Equation (3.5) allows for mini-batch training and stochastic optimization to efficiently estimate $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$ from the data (Ma *et al.*, 2019). Given, $Q_{\boldsymbol{\omega}}(\mathbf{a})$, it is straightforward to make predictions. A limitation is, however, that the predictive distribution for y is Gaussian in regression. More precisely, given a test point $\mathbf{x}^{\star}$,

$$Q(\mathbf{f}^{\star}) = \mathcal{N}\big(m(\mathbf{x}^{\star}) + \phi(\mathbf{x}^{\star})^{T}\boldsymbol{m}\,, \boldsymbol{\phi}(\mathbf{x}^{\star})^{T}\boldsymbol{S}\boldsymbol{\phi}(\mathbf{x}^{\star})\big)\,, \tag{3.6}$$

which is straightforward given the linear approximation $f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^{\mathrm{T}}\mathbf{a} + m^{\star}(\mathbf{x})$ and the variational distribution $Q_{\omega}(\mathbf{a}) \sim \mathcal{N}(\boldsymbol{m}, \boldsymbol{S})$. Using a Gaussian likelihood $P(y \mid \mathbf{f}) = \mathcal{N}(0, \sigma^2)$, leads to the predictive distribution

$$\begin{aligned} P(y^{\star}|\mathbf{x}^{\star}) &= \int P(y^{\star}|\mathbf{f}^{\star})Q(\mathbf{f}^{\star}) \\ &= \mathcal{N}\big(m(\mathbf{x}^{\star}) + \phi(\mathbf{x}^{\star})^T\boldsymbol{m}\,, \boldsymbol{\phi}(\mathbf{x}^{\star})^T\boldsymbol{S}\boldsymbol{\phi}(\mathbf{x}^{\star}) + \sigma^2\big)\,. \end{aligned} \tag{3.7}$$

The main drawback is that Gaussian distributions may not be flexible enough to explain complex data patterns.

3.2 COMPUTATIONAL COMPLEXITY

If not for the linear regression approximation, the computational complexity of VIPs would match that of standard GPs, being cubic on training and quadratic on prediction. However, this approach is inapplicable to datasets with thousands of points.

The usage of the linear approximation simplifies the training complexity using stochastic optimization and mini-batches to be $\mathcal{O}(BS^2D + BSC)$ where B denotes the size of the mini-batch, S the number of prior samples, D the data targets dimensionality and C the cost of taking a prior sample. In most cases, the second term (linear on S) would be much smaller than the quadratic term. However, given that in practise S is relatively small ($S = 20$, Ma *et al.* (2019)), using more complex prior functions would lead to the linear term being the most significant.

3.3 IPS GENERALIZE GPS

From their sole definition, it is clear that implicit processes generalize the concept of Gaussian processes. More over, letting $P_{\mathbf{z}} = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $g_{\boldsymbol{\theta}}(\mathbf{x}_n, \mathbf{z}) = \boldsymbol{B}(\mathbf{x})^T\mathbf{z}$ with $\boldsymbol{B}(\mathbf{x})$ the Cholesky decomposition of a kernel evaluated in $\mathbf{x}$, the definition of Gaussian process is recovered. Given this, it is intuitive that IPs have the flexibility to give better estimations, tuning its prior parameters to their observed data. This is shown in Ma *et al.* (2019), where VIPs are employed on different regression datasets, outperforming GPs in both synthetic and real datasets, and on estimating uncertainty in missing sections of time-series. Moreover, the obtained results showed that VIPs significantly improved results on molecule regression data (Ma *et al.*, 2019).

3.4 GAUSSIAN KULLBACK-LEIBLER

Let us derive here the KL between two Gaussians, since it has a closed form solution and will be of use in later sections of the text. Let P and Q be two k-dimensional Gaussian distributions over the same variable $\mathbf{a}$. Namely,

$$P(\mathbf{a}) = \mathcal{N}(\mathbf{a}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \quad \text{and} \quad Q(\mathbf{a}) = \mathcal{N}(\mathbf{a}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)\,. \tag{3.8}$$

The Kullback-Leibler divergence between both distributions can be analytically computed as

$$KL\Big(P(\mathbf{a}) \mid Q(\mathbf{a})\Big) = \mathbb{E}_{P(\mathbf{a})}\Big[\log P(\mathbf{a}) - \log Q(\mathbf{a})\Big]\,. \tag{3.9}$$

First of all, the logarithm of the Gaussian density can be rewritten in terms of the trace of a matrix:

$$\begin{aligned}\log P(\mathbf{a}) &= -\frac{k}{2}\log(2\pi) - \log\det(\boldsymbol{\Sigma}_1) - \frac{1}{2}\left((\mathbf{a}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_1^{-1}(\mathbf{a}-\boldsymbol{\mu}_1)\right) \\ &= -\frac{k}{2}\log(2\pi) - \log\det(\boldsymbol{\Sigma}_1) - \frac{1}{2}\text{tr}\left(\boldsymbol{\Sigma}_1^{-1}(\mathbf{a}-\boldsymbol{\mu}_1)(\mathbf{a}-\boldsymbol{\mu}_1)^T\right)\,.\end{aligned} \tag{3.10}$$

Given that the trace is a linear operator, its expectation is simplified,

$$\begin{aligned}\mathbb{E}_{P(\mathbf{a})}\Big[\text{tr}\left(\boldsymbol{\Sigma}_1^{-1}(\mathbf{a}-\boldsymbol{\mu}_1)(\mathbf{a}-\boldsymbol{\mu}_1)^T\right)\Big] &= \text{tr}\left(\boldsymbol{\Sigma}_1^{-1}\mathbb{E}_{P(\mathbf{a})}\Big[(\mathbf{a}-\boldsymbol{\mu}_1)(\mathbf{a}-\boldsymbol{\mu}_1)^T\Big]\right) \\ &= \text{tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_1\right) = k\,.\end{aligned} \tag{3.11}$$

On the other hand, the corresponding term over Q is

$$\begin{aligned}\mathbb{E}_{P(\mathbf{a})}\Big[\text{tr}\left(\boldsymbol{\Sigma}_2^{-1}(\mathbf{a}-\boldsymbol{\mu}_2)(\mathbf{a}-\boldsymbol{\mu}_2)^T\right)\Big] &= \text{tr}\left(\boldsymbol{\Sigma}_2^{-1}\mathbb{E}_{P(\mathbf{a})}\Big[(\mathbf{a}-\boldsymbol{\mu}_2)(\mathbf{a}-\boldsymbol{\mu}_2)^T\Big]\right) \\ &= \text{tr}\left(\boldsymbol{\Sigma}_2^{-1}\mathbb{E}_{P(\mathbf{a})}\Big[\mathbf{a}\mathbf{a}^T - 2\mathbf{a}\boldsymbol{\mu}_2^T + \boldsymbol{\mu}_2\boldsymbol{\mu}_2^T\Big]\right) \\ &= \text{tr}\left(\boldsymbol{\Sigma}_2^{-1}\left(\boldsymbol{\Sigma}_1 + \boldsymbol{\mu}_1\boldsymbol{\mu}_1^T - 2\boldsymbol{\mu}_1\boldsymbol{\mu}_2^T + \boldsymbol{\mu}_2\boldsymbol{\mu}_2^T\right)\right)\,,\end{aligned} \tag{3.12}$$

which can be re-arranged as

$$\begin{aligned}\text{tr}\left(\boldsymbol{\Sigma}_2^{-1}\left(\boldsymbol{\Sigma}_1 + \boldsymbol{\mu}_1\boldsymbol{\mu}_1^T - 2\boldsymbol{\mu}_1\boldsymbol{\mu}_2^T + \boldsymbol{\mu}_2\boldsymbol{\mu}_2^T\right)\right) &= \text{tr}\left(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right) + \text{tr}\left(\boldsymbol{\Sigma}_2^{-1}\left(\boldsymbol{\mu}_1\boldsymbol{\mu}_1^T - 2\boldsymbol{\mu}_1\boldsymbol{\mu}_2^T + \boldsymbol{\mu}_2\boldsymbol{\mu}_2^T\right)\right) \\ &= \text{tr}\left(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right) + \boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_1 - 2\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2 + \boldsymbol{\mu}_2^T\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2 \\ &= \text{tr}\left(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right) + (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)\,.\end{aligned} \tag{3.13}$$

Using all the derived terms, the Kullback-Leibler divergence can be writen as

$$KL\Big(P(\mathbf{a}) \;\Big|\; Q(\mathbf{a})\Big) = \log\left(\frac{\det(\boldsymbol{\Sigma}_2)}{\det(\boldsymbol{\Sigma}_1)}\right) + \text{tr}\left(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1\right) + (\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)^T\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) - k\,. \tag{3.14}$$

Moreover, the specific case when $\boldsymbol{\mu}_2 = 0$ and $\boldsymbol{\Sigma}_2 = \boldsymbol{I}$, simplifies the above equality as

$$KL\Big(P(\mathbf{a}) \;\Big|\; \mathcal{N}(0,\boldsymbol{I})\Big) = -\log\det(\boldsymbol{\Sigma}_1) + \text{tr}\left(\boldsymbol{\Sigma}_1\right) + \boldsymbol{\mu}_1^T\boldsymbol{\mu}_1 - k\,. \tag{3.15}$$

3.5 GAUSSIAN EXPECTATION

In this section, the expectation on Equation (3.5) is shown to be possible to be computed in closed form using a Gaussian likelihood (as it is used in the experiments). Namely, consider $Q(f) = \mathcal{N}(\mu, \Sigma)$, $P(y|f) = \mathcal{N}(y|f, \sigma^2)$ and $\alpha \in (0, 1)$.

First of all, $P(y|f)^\alpha$ can be shown to be an unnormalized Gaussian:

$$P(y|f)^\alpha = \left(\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\frac{(y-f)^2}{\sigma^2}}\right)^\alpha = \frac{1}{(2\pi\sigma^2)^{\frac{\alpha}{2}}}e^{-\frac{1}{2}\frac{(y-f)^2}{\sigma^2/\alpha}}\,. \tag{3.16}$$

From this expression, it is straightforward to consider a Gaussian distribution of a specific variance, $\mathcal{N}(y|f, \frac{\sigma^2}{\alpha})$, such that,

$$P(y|f)^\alpha = (2\pi\sigma^2/\alpha)^{\frac{1}{2}}\frac{1}{(2\pi\sigma^2)^{\frac{\alpha}{2}}}\mathcal{N}(y|f, \sigma^2/\alpha)\,. \tag{3.17}$$

Using this unnormalized Gaussian, its expectation under $Q(f)$ is

$$\begin{aligned}\mathbb{E}_{Q(f)}\Big[P(y|f)^\alpha\Big] &= \mathbb{E}_{Q(f)}\left[(2\pi\sigma^2/\alpha)^{\frac{1}{2}}\frac{1}{(2\pi\sigma^2)^{\frac{\alpha}{2}}}\mathcal{N}(y|f,\sigma^2/\alpha)\right]\\ &= (2\pi\sigma^2/\alpha)^{\frac{1}{2}}\frac{1}{(2\pi\sigma^2)^{\frac{\alpha}{2}}}\mathbb{E}_{Q(f)}\left[\mathcal{N}(y|f,\sigma^2/\alpha)\right]\,.\end{aligned}$$

Finally, using that the convolution of Gaussians is Gaussian raises the following expression,

$$\mathbb{E}_{Q(f)}\Big[P(y|f)^\alpha\Big] = (2\pi\sigma^2/\alpha)^{\frac{1}{2}}\frac{1}{(2\pi\sigma^2)^{\frac{\alpha}{2}}}\mathcal{N}(y|\mu,\Sigma+\sigma^2/\alpha)\,. \tag{3.18}$$

4

RELATED RESEARCH

The relationship between GPs and IPs, such as BNNs, has been extensively studied recently. A single-layer BNN with cosine activations and infinite width is equivalent to a GP with RBF kernel (Hensman *et al.*, 2017). A deep BNN is equivalent to a GP with a compositional kernel (Cho & Saul, 2009), as shown in Lee *et al.* (2018). These methods make it possible to create expressive kernels for GPs. An inverse reasoning is used in Flam-Shepherd *et al.* (2017) where the properties of a GP prior are encoded into the weight priors of a BNN.

As described in Section 3, VIPs (Ma *et al.*, 2019) raises from the treatment of BNNs as instances of IPs. For this, an approximate GP is used to assist inference. Specifically, a prior GP is built with mean and covariance function given by the prior IP, a BNN. VIPs can make use of the more flexible IP prior, whose parameters can be inferred from the data, resulting in improved results over GPs (Ma *et al.*, 2019). However, it is limited by the fact that VIP's predictive distribution is Gaussian, as a consequence of the GP approximation. Deep variational implicit processes (DVIPs) overcome this problem providing a non-Gaussian predictive distribution. It is also expected to lead to a more flexible model with better calibrated uncertainty estimates.

Besides VIP there are other methods that have tried to make inference using IPs. In Sun *et al.* (2019) *functional Bayesian neural networks* (fBNN) are introduced, where a second IP is used to approximate the posterior of the first IP. This is a more flexible approximation than that of VIP. However, because both the prior and the posterior are implicit, the noisy gradient of the variational ELBO is intractable and has to be approximated. For this, a spectral gradient estimator is used (Shi *et al.*, 2018). To ensure that the posterior IP resembles the prior IP in data-free regions, fBNN relies on uniformly covering the input space. In high-dimensional spaces this can lead to poor results. As a consequence of the spectral gradient estimator, fBNN cannot tune the prior IP parameters to the data. In the particular case of a GP prior, fBNN simply maximizes the marginal likelihood of the GP w.r.t. the prior parameters. However, a GP prior implies a GP posterior. This questions using a second IP for posterior approximation.

Sparse implicit processes (SIPs) consider an inducing point based approach for approximate inference in the context of IPs (Santana *et al.*, 2021). SIP does not have the limitations of neither VIP nor fBNN, given that it produces flexible predictive distributions (Gaussian mixtures) and it can adjust the prior parameters to the data. SIP, however, relies on a classifier to estimate the KL-term in the variational ELBO, which adds computational cost. SIP's improvements over VIP are orthogonal to those of DVIP over VIP and, in principle, SIP could also be used as the building blocks of DVIP, leading to even better results.

Functional variational inference (FVI, Ma & Hernández-Lobato (2021)) minimizes the KL-divergence between stochastic process for approximate inference using IPs. Specifically, between the model's IP posterior and a second IP, as fBNN. This is done efficiently by approximating first the IP prior using a stochastic process generator (SPG). Then, a second SPG is used to efficiently approximate the posterior of the previous SPG.Both SPGs share key features that make this an easy task. However, FVI is also limited, as fBNN, in the sense that it cannot adjust the prior to the data, which questions its practical utility.

As shown in Santana *et al.* (2021), adjusting the prior IP to the observed data is key for accurate predictions. This questions using fBNN and FVI as the building blocks of a model using deep IP priors on the target function. Moreover, these methods do not consider deep architectures such as the one shown in Figure 13. For this reason, comparative results focus on comparing with VIP, as DVIP generalizes VIP.

The concatenation of IPs with the goal of describing priors over functions has not been studied previously in the literature. However, the concatenation of GPs, following a multi-layer architecture and resulting in deep GPs (DGPs), has received a lot of attention from the community (Lawrence & Moore, 2007; Bui *et al.*, 2016; Cutajar *et al.*, 2017; Salimbeni & Deisenroth, 2017; Havasi *et al.*, 2018; Yu *et al.*, 2019). In principle, DVIP is a generalization of DGPs in the same way as IPs generalize GPs. Namely, the IP prior of each layer's unit can simply be a GP. Samples from such a GP prior can be efficiently obtained using, *e.g.*, a 1-layer BNN with cosine activation functions that is wide enough (Rahimi & Recht, 2007; Cutajar *et al.*, 2017). DGPs are hence restricted to GP priors, while DVIP has the extra flexibility of considering a wider range of IP priors that need not be GPs. As a matter of fact, in our experiments, DVIP significantly outperforms DGPs in image-related datasets, where using specific IP priors based on convolutional neural networks, can give a significant advantage. Experiments also show that DVIP is faster to train than a DGP, and the difference becomes larger with the number of layers L.

Part II

DEEP VARIATIONAL IMPLICIT PROCESS

In this part, *deep variational implicit processes* are introduced, showing the approximated learning framework using Monte-Carlo samples to bypass the exact model intractability.

Details on specific techniques such as input propagation through the layers and the models full computational complexity are included.

Several experiments have being conducted, both in regression and classification datasets, from hundreds to millions of data-points.

Moreover, some of the experiments involve the usage of domain specific such as convolutional networks, and regression results over missing gaps of time series.

5

DEFINITION

Despite the fact that sparse approaches for Gaussian processes addressed their computational difficulty, GPs remain inapplicable to many situations due to their kernel functions. Commonly used kernels have simple similarity metrics that do not fit for every dataset or even different regions of the same dataset. Layering GPs in the same way as multi-layer perceptrons (*deep Gaussian processes*, Damianou & Lawrence (2013)), solves this issue, at the cost of making them highly non-linear and intractable to analytical solutions. Moreover, deep GP's predictive distribution does not have to be Gaussian. Different techniques have being applied in order to handle this issues, such as approximated expectation propagation (Bui *et al.*, 2016) and doubly stochasticity via Monte Carlo sampling (Salimbeni & Deisenroth, 2017).

Deep variational implicit processes (DVIPs) are models that consider a deep implicit process as the prior for the latent function. A deep implicit process is a concatenation of multiple IPs, recursively defining an implicit prior over latent functions. Figure 13 illustrates the architecture considered, which is fully connected. The prior on the function at layer l and unit h, $f_h^l(\cdot)$, is an independent IP whose inputs are given by the outputs of the previous layer. Let H_l be the l-th layer dimensionality.

Definition 5.1 (Deep Implicit Process). A deep implicit process is a collection of random variables $\{f_{h,n}^l : l = 1, \dots, L \wedge h = 1 \dots, H_l \wedge n = 1, \dots, N\}$ such that each $f_{h,n}^l = f_h^l(\mathbf{f}_{\cdot,n}^{l-1})$, with $\mathbf{f}_{\cdot,n}^{l-1}$ the output of the previous layer in the network, i.e., $\mathbf{f}_{\cdot,n}^{l-1} = (f_{1,n}^{l-1}, \dots, f_{H_{l-1},n}^{l-1})^{\mathrm{T}}$, and each $f_h^l(\cdot)$ is an independent IP:

$$f_h^l(\cdot) \sim \mathcal{IP}\left(g_{\boldsymbol{\theta}_h^l}(\cdot, \boldsymbol{z}), P_{\boldsymbol{z}}\right), \tag{5.1}$$

where $\mathbf{f}_{\cdot,n}^0 = \mathbf{x}_n$ symbolizes the initial input features to the multi-layer IP network.

As in VIPs, GP approximations are considered for all the IPs in the deep IP prior defined above. These GPs are further approximated using a linear model, as in VIPs. This provides an expression for $f_{h,n}^l$ given the previous layer's output $\mathbf{f}_{\cdot,n}^{l-1}$ and $\mathbf{a}_h^l$, the coefficients of the linear model for the unit h at layer l. Namely,

$$f_{h,n}^l = \boldsymbol{\phi}_h^l(\mathbf{f}_{\cdot,n}^{l-1})^{\mathrm{T}}\mathbf{a}_h^l + m_{h,l}^\star(\mathbf{f}_{\cdot,n}^{l-1}), \tag{5.2}$$

where $\boldsymbol{\phi}_h^l(\cdot)$ and $m_{h,l}^\star(\cdot)$ depend on the prior IP parameters $\boldsymbol{\theta}_h^l$. To increase the flexibility of the model, latent Gaussian noise is added around each inner $f_{h,n}^l$ with variance $\sigma_{l,h}^2$ for $l \in \{1, \dots, L-1\}$, that is, all inner layers. From now on $\boldsymbol{\sigma}_l^2$ will denote the set of unit noises

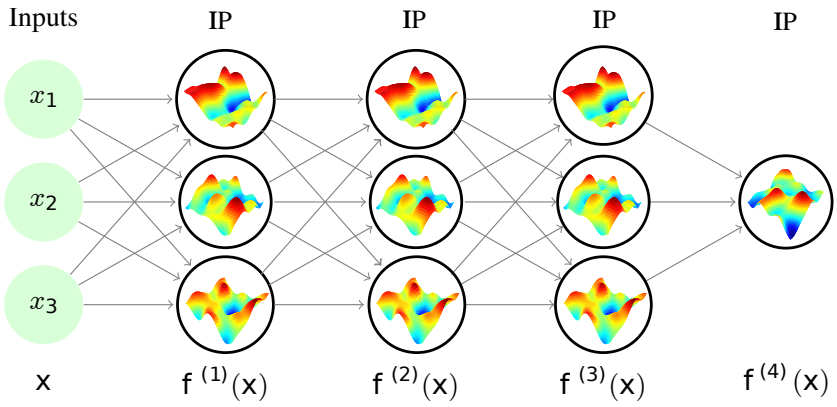


Figure 13: Deep VIP in which the input to an IP is the output of a previous IP. A fully connected architecture is considered.

at layer l, thus, $\boldsymbol{\sigma}_l^2 = \{\sigma_{l,h}^2\}_{h=1}^{H_l}$. As a result of these approaches, $P(f_{h,n}^l|\mathbf{f}_{:,n}^{l-1}, \mathbf{a}_h^l)$ is Gaussian with a linear mean in terms of $\mathbf{a}_h^l$

$$P(f_{h,n}^l|\mathbf{f}_{:,n}^{l-1}, \mathbf{a}_h^l) = \mathcal{N}\left(f_{h,n}^l|\, m_{h,l}^\star(\mathbf{f}_{:,n}^{l-1}) + \boldsymbol{\phi}_h^l(\mathbf{f}_{:,n}^{l-1})^{\mathrm{T}}\mathbf{a}_h^l\,, \sigma_{l,h}^2\right) \quad \forall l \in \{1,\ldots,L-1\} \tag{5.3}$$

and linearly deterministic for the last layer

$$f_{h,n}^L = m_{h,L}^\star(\mathbf{f}_{:,n}^{L-1}) + \boldsymbol{\phi}_h^L(\mathbf{f}_{:,n}^{L-1})^{\mathrm{T}}\mathbf{a}_h^L\,. \tag{5.4}$$

Let $\mathbf{A}^l = \{\mathbf{a}_1^l, \ldots, \mathbf{a}_{H_l}^l\}$ and $\mathbf{F}^l = \{\mathbf{f}_{:,1}^l, \ldots, \mathbf{f}_{:,N}^l\}$. The joint distribution of all the variables (observed and latent) in DVIP is

$$P\left(\mathbf{y}, \{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L\right) = \underbrace{\prod_{n=1}^N P(y_n|\mathbf{f}_{:,n}^L)}_{\text{Likelihood}} \underbrace{\prod_{n=1}^N \prod_{l=1}^L \prod_{h=1}^{H_l} P(f_{h,n}^l|\mathbf{f}_{:,n}^{l-1}, \mathbf{a}_h^l)P(\mathbf{a}_h^l)}_{\text{DVIP prior}}\,, \tag{5.5}$$

where the prior for the linear coefficients is set to a standard Gaussian, i.e, $P(\mathbf{a}_h^l) = \mathcal{N}(\mathbf{a}_h^l|\mathbf{0}, \mathbf{I})$. It may seem that the prior assumes independence across points. However, dependencies raise by marginalizing out each $\mathbf{a}_h^l$, which is tractable since the model is linear in $\mathbf{a}_h^l$ (this marginalization can be seen as a convolution of Gaussians).

The posterior over the latent variables $P\left(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L|\mathbf{y}\right)$ is approximated using a similar approximation to that of deep GPs (Salimbeni & Deisenroth, 2017). This approximation has a fixed part and a tunable part, simplifying dependencies among layer units, but maintaining dependencies between layers:

$$Q(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L) = \prod_{n=1}^N \prod_{l=1}^L \prod_{h=1}^{H_l} P(f_{h,n}^l|\mathbf{f}_n^{l-1}, \mathbf{a}_h^l)Q(\mathbf{a}_h^l)\,, \qquad Q(\mathbf{a}_h^l) = \mathcal{N}(\mathbf{a}_h^l|\boldsymbol{m}_h^l, \boldsymbol{S}_h^l)\,, \tag{5.6}$$

where the factors $P(f_{h,n}^l|\mathbf{f}_n^{l-1}, \mathbf{a}_h^l)$ are fixed to be the same factors as those in Equation (5.5), and the factors $Q(\mathbf{a}_d^l)$ are the ones being specifically tuned. Computing $Q(\mathbf{f}_{:,n}^L)$, as specified by Equation (5.6) is intractable. However, one can easily sample from $Q(\mathbf{f}_{:,n}^L)$, as described next, by propagating samples through the network (Section 6.1).

Using Equation (5.6), a variational lower bound can be derived; at whose maximum the Kullback-Leibler (KL) divergence between $Q(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L)$ and $P\left(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L|\mathbf{y}\right)$ is minimized (see Section 7 for details). Namely the variational objective is,

$$\mathcal{L}\left(\Omega, \Theta, \{\boldsymbol{\sigma}_l^2\}_{l=1}^{L-1}\right) = \sum_{n=1}^{N} \mathbb{E}_{Q(\mathbf{f}_{:,n}^L)}\left[\log P\left(y_n|\mathbf{f}_{:,n}^L\right)\right] - \sum_{l=1}^{L}\sum_{h=1}^{H_l} \mathrm{KL}\left(Q(\mathbf{a}_h^l) \mid P(\mathbf{a}_h^l)\right). \tag{5.7}$$

In the above formula, $\Omega = \{\boldsymbol{m}_h^l, \boldsymbol{S}_h^l : l = 1, \ldots, L \wedge h = 1, \ldots, H_l\}$ are the parameters of Q and $\Theta = \{\boldsymbol{\theta}_h^l : l = 1, \ldots, L \wedge h = 1, \ldots, H_l\}$ are the deep IP prior parameters. $\mathrm{KL}(Q(\mathbf{a}_h^l)|P(\mathbf{a}_h^l))$ involves Gaussian distributions and can be evaluated analytically (Section 3.4). The expectations $\mathbb{E}_{Q(\mathbf{f}_{:,n}^L)}\left[\log P(y_n|\mathbf{f}_{:,n}^L)\right]$ are intractable. However, they can be approximated via Monte Carlo, using the reparametrization trick (Kingma & Welling, 2014) (Section 6.1). Moreover, $\sum_{n=1}^N \mathbb{E}_{Q(\mathbf{f}_{:,n}^L)}\left[\log P(y_n|\mathbf{f}_{:,n}^L)\right]$ can be approximated using mini-batches of size B:

$$\mathcal{L}\left(\Omega, \Theta, \{\boldsymbol{\sigma}_l^2\}_{l=1}^{L-1}\right) \approx \frac{N}{B}\sum_{b=1}^{B} \mathbb{E}_{Q(\mathbf{f}_{:,b}^L)}\left[\log P\left(y_b|\mathbf{f}_{:,b}^L\right)\right] - \sum_{l=1}^{L}\sum_{h=1}^{H_l} \mathrm{KL}\left(Q(\mathbf{a}_h^l) \mid P(\mathbf{a}_h^l)\right). \tag{5.8}$$

The consequence is that Equation (5.8) can be maximized w.r.t. Ω, Θ and $\{\boldsymbol{\sigma}_l^2\}_{l=1}^{L-1}$, using stochastic optimization methods. Maximization w.r.t. Θ allows for prior adaptation to the observed data, which is a key factor when considering IP priors.

6

MODEL DETAILS

6.1 SAMPLING FROM THE MARGINAL POSTERIOR APPROXIMATION

The approximation of the variational ELBO in Equation (5.8) requires samples from $Q(\mathbf{f}_{:,n}^L)$ for all the instances in a mini-batch. This marginal only depends on the variables of the inner layers and units $f_{h,n}^l$corresponding to the n-th instance (see Section 7 for the theoretical derivation). Therefore, samples from $Q(\mathbf{f}_{:,n}^L)$ can be drawn by recursively propagating samples from the first to the last layer, using $\mathbf{x}_n$ as the network input. Specifically, $Q(f_{h,n}^l|\mathbf{f}_{:,n}^{l-1})$ is Gaussian

$$Q(f_{h,n}^l|\mathbf{f}_{:,n}^{l-1}) = \int P(f_{h,n}^l|\mathbf{f}_{:,n}^{l-1}, \mathbf{a}_h^l) Q(\mathbf{a}_h^l)\, \mathrm{d}\mathbf{a}_h^l = \mathcal{N}\left(f_{h,n}^l|\, \hat{m}_{h,n}^l(\mathbf{f}_{:,n}^{l-1}), \hat{v}_{h,n}^l(\mathbf{f}_{:,n}^{l-1})\right) . \tag{6.1}$$

with mean and variance:

$$\begin{aligned} \hat{m}_{h,n}^l(\mathbf{f}_{:,n}^{l-1}) &= \boldsymbol{\phi}_h^l(\mathbf{f}_{:,n}^{l-1})^{\mathrm{T}} \boldsymbol{m}_h^l + m_{h,l}^\star(\mathbf{f}_{:,n}^{l-1}) , \\ \hat{v}_{h,n}^l(\mathbf{f}_{:,n}^{l-1}) &= \boldsymbol{\phi}_h^l(\mathbf{f}_{:,n}^{l-1})^{\mathrm{T}} \boldsymbol{S}_h^l \boldsymbol{\phi}_h^l(\mathbf{f}_{:,n}^{l-1}) + \sigma_{l,h}^2 \mathbb{I}[l \neq L] , \end{aligned} \tag{6.2}$$

where $\boldsymbol{m}_h^l$ and $\boldsymbol{S}_h^l$ are the parameters of $Q(\mathbf{a}_h^l)$.

Initially, consider $l = 1$. We set $\hat{\mathbf{f}}_{:,n}^0 = \mathbf{x}_n$ and generate a sample from $Q(f_{h,n}^l|\hat{\mathbf{f}}_{:,n}^0)$ for $h = 1, \ldots, H_l$. Let $\hat{\mathbf{f}}_{:,n}^l$ be that sample. Then, use $\hat{\mathbf{f}}_{:,n}^l$ as the input for the next layer. This process repeats for $l = 2, \ldots, L$, until obtaining $\hat{\mathbf{f}}_{:,n}^L \sim Q(\mathbf{f}_{:,n}^L)$. Algorithm 1 shows the full procedure of training the model.

6.2 MAKING PREDICTIONS FOR NEW INSTANCES

Let $\mathbf{f}_{:,\star}^L$ be the values at the last layer for the new instance $\mathbf{x}_\star$. The distribution $Q(\mathbf{f}_{:,\star}^L)$ is approximated by propagating R Monte Carlo samples through the network. Then,

$$Q(\mathbf{f}_{:,\star}^L) \approx \frac{1}{R} \sum_{r=1}^{R} \prod_{h=1}^{H_L} \mathcal{N}\left(f_{h,\star}^L|\hat{m}_{h,\star}^L(\hat{\mathbf{f}}_{:,\star}^{L-1,r}), \hat{v}_{h,\star}^L(\hat{\mathbf{f}}_{:,\star}^{L-1,r})\right) , \tag{6.3}$$

where $\hat{\mathbf{f}}_{:,\star}^{L-1,r}$ is the r-th sample arriving at layer $L-1$ and $\hat{m}_{h,\star}^L(\cdot)$ and $\hat{v}_{h,\star}^L(\cdot)$ are given by Equation (6.2). Notice that Equation (6.3) is a mixture of R Gaussians, which is expected to be more flexible than the predictive distribution of VIPs. In the conducted experiments R is set to 100 for testing and to 1 for training. Computing, $P(y_\star) = \mathbb{E}_Q[P(y_\star|\mathbf{f}_{:,\star}^L)]$ is tractable

Algorithm 1: Training Deep Variational Implicit Process

Data: Set of training data $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ and number of iterations n_iter.
Sample random noise for each layer's prior $\mathbf{z} \sim P_{\mathbf{z}}$;
for $i \in \{1, \ldots, n\_iter\}$ **do**
  /* Make Monte-Carlo resamples */
  $\mathbf{X}' \leftarrow (\mathbf{X}, \ldots, \mathbf{X})$;
  $\mathbf{f}^0 \leftarrow \mathbf{X}'$;
  for $l \in \{1, \ldots, L\}$ **do**
    /* Get prior samples */
    $\boldsymbol{\phi}_l \leftarrow f_l(\mathbf{X}')$;
    /* Sample from marginal distribution */
    for $n \in \{1, \ldots, N\}$ **do**
      $\mathbf{f}^l_{\cdot,n} \leftarrow Q(\mathbf{f}^l_{\cdot,n} | \mathbf{f}^{l-1}_{\cdot,n})$;
  /* Compute approximated expected log-likelihood */
  $ll \leftarrow \sum_{n=1}^{N} \mathbb{E}_{Q(\mathbf{f}^L_{\cdot,n})}\big[\log P\big(y_n | \mathbf{f}^L_{\cdot,n}\big)\big] \approx \sum_{n=1}^{N} \log P\big(y_n | \mathbf{f}^L_{\cdot,n}\big)$;
  /* Initialize KL */
  $kl \leftarrow 0$;
  for $l \in \{1, \ldots, L\}$ **do**
    /* Add layer KL */
    $kl \leftarrow kl + KL(Q(\mathbf{a}^l) | P(\mathbf{a}^l))$;
  /* Compute ELBO */
  $\mathcal{L} \leftarrow ll - kl$;
  /* Optimize parameters */
  $\boldsymbol{\Theta} \leftarrow Optimizer(\boldsymbol{\Theta}, \mathcal{L})$;

in regression, and can be approximated using 1-dimensional quadrature in binary and multi-class classification, as in the case of DGPs (Salimbeni & Deisenroth, 2017). Assume a Gaussian likelihood for regression problems, $P(y_\star | \mathbf{f}^L_{\cdot,\star}) = \mathcal{N}(y_\star | \mathbf{f}^L_{\cdot,\star}, \boldsymbol{\sigma}^2_L \boldsymbol{I})$, where $\boldsymbol{\sigma}^2_L = (\sigma^2_{L,1}, \ldots, \sigma^2_{L,H_L})^T$, the expectation takes the form:

$$\begin{aligned} P(y_\star | \mathbf{x}_\star) &= \mathbb{E}_{Q(\mathbf{f}^L_{\cdot,\star})}[P(y_\star | \mathbf{f}^L_{\cdot,\star})] = \int Q(\mathbf{f}^L_{\cdot,\star}) P(y_\star | \mathbf{f}^L_{\cdot,\star}) \, d\mathbf{f}^L_{\cdot,\star} \\ &= \frac{1}{R} \sum_{r=1}^{R} \prod_{h=1}^{H_L} \mathcal{N}\Big(f^L_{h,\star} | \hat{m}^L_{h,\star}(\hat{\mathbf{f}}^{L-1,r}_{\cdot,\star}) \, , \hat{v}^L_{h,\star}(\hat{\mathbf{f}}^{L-1,r}_{\cdot,\star}) + \sigma^2_{L,h}\Big) \, . \end{aligned} \tag{6.4}$$

Remark 1. Using a Gaussian Likelihood is equivalent to consider Gaussian noise at the final layer similarly to the noise employed in the inner layers. The noise on the Gaussian likelihood is denoted as $\boldsymbol{\sigma}^2_L$ to highlight this fact.

6.3 INPUT PROPAGATION

Inspired by the *skip layer* approach of deep convolutional networks, *e.g.* ResNet (He *et al.*, 2016), and the addition of the original input to each layer on DGPs (Duvenaud *et al.*, 2014; Salimbeni & Deisenroth, 2017), the same approach is implemented on DVIPs. For this, the

previous input is added to the mean in Equation (6.2) if the input and output dimension of the layer is the same, except in the last layer where the mean does not change. In short,

$$\hat{m}^l_{h,n}(\mathbf{f}^{l-1}_{\cdot,n}) = \boldsymbol{\phi}^l_h(\mathbf{f}^{l-1}_{\cdot,n})^{\mathrm{T}}\boldsymbol{m}^l_h + m^\star_{h,l}(\mathbf{f}^{l-1}_{\cdot,n}) + f^{l-1}_{h,n}\mathbb{I}[l \neq L] \quad \forall l = 1, \dots, L\,. \tag{6.5}$$

6.4 COMPUTATIONAL COMPLEXITY

The computational cost at layer l in DVIPs is $\mathcal{O}(BS^2H_l + BSC_l)$, where S is the number of samples from the prior IPs, B is the size of the mini-batch, H_l is the output dimension of the layer and C_l is the cost of sampling from the prior. Notice the same number of prior samples is used for all the layers for simplicity. Moreover, only the diagonal of the covariance matrices is being computed as in VIPs.

The first term of the cost is similar to that of a DGPs, which have a squared cost in terms of the number of inducing points (Hensman *et al.*, 2013; Bui *et al.*, 2016; Salimbeni & Deisenroth, 2017), notice that in that case, only the diagonal of the covariance matrix is computed, otherwise, the cost would be cubic. However, the number of prior IP samples S is smaller than the typical number of inducing points in DGPs. In fact, the experiments carried out $S = 20$, as suggested for VIPs (Ma *et al.*, 2019). Considering a DVIP with L layers, the total cost is in $\mathcal{O}(BS^2(H_1 + \cdots + H_L) + BS(C_1 + \cdots + C_L))$. Where the second term is negligible unless heavy models are used as prior. Experiments show that, despite the generation of the prior IP samples using BNNs, DVIPs are faster compared to DGPs, and the gap between the two becomes bigger as L increases.

7

FURTHER DERIVATIONS

DERIVATION OF THE ELBO

In this section, the variational ELBO for DVIPs is derived in closed form. Consider the general definition of the ELBO in variational inference, i.e

$$\mathcal{L}\left(\Omega, \Theta, \boldsymbol{\sigma}_l^2\}_{l=1}^{L-1}\right) = \mathbb{E}_{Q(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L)}\left[\log \frac{P\left(\mathbf{y}, \{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L\right)}{Q(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L)}\right], \tag{7.1}$$

using DVIPs model specification:

$$\begin{aligned} P\left(\mathbf{y}, \{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L\right) &= \prod_{n=1}^{N} P(y_n \mid \mathbf{f}_{:,n}^L) \prod_{n=1}^{N}\prod_{l=1}^{L}\prod_{h=1}^{H_l} P(f_{h,n}^l \mid \mathbf{a}_h^l) P(\mathbf{a}_h^l), \\ Q(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L) &= \prod_{n=1}^{N}\prod_{l=1}^{L}\prod_{h=1}^{H_l} P(f_{h,n}^l \mid \mathbf{a}_h^l) Q(\mathbf{a}_h^l), \end{aligned} \tag{7.2}$$

the variational lower bound takes the following form:

$$\begin{aligned} \mathcal{L} &= \mathbb{E}_{Q(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L)}\left[\log \frac{P\left(\mathbf{y}, \{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L\right)}{Q(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L)}\right] \\ &= \mathbb{E}_{Q(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L)}\left[\log \frac{\prod_{n=1}^N P(y_n \mid \mathbf{f}_{:,n}^L) \prod_{n=1}^N \prod_{l=1}^L \prod_{h=1}^{H_l} \cancel{P(f_{h,n}^l \mid \mathbf{a}_h^l)} P(\mathbf{a}_h^l)}{\prod_{n=1}^N \prod_{l=1}^L \prod_{h=1}^{H_l} \cancel{P(f_{h,n}^l \mid \mathbf{a}_h^l)} Q(\mathbf{a}_h^l)}\right] \\ &= \mathbb{E}_{Q(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L)}\left[\log \frac{\prod_{n=1}^N P(y_n \mid \mathbf{f}_{:,n}^L) \prod_{l=1}^L \prod_{h=1}^{H_l} P(\mathbf{a}_h^l)}{\prod_{l=1}^L \prod_{h=1}^{H_l} Q(\mathbf{a}_h^l)}\right]. \end{aligned} \tag{7.3}$$

From this expression, the expectation can be split in two terms

$$\mathcal{L} = \mathbb{E}_{Q(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L)}\left[\log \prod_{n=1}^{N} P(y_n \mid \mathbf{f}_{:,n}^L)\right] + \mathbb{E}_{Q(\{\mathbf{F}^l, \mathbf{A}^l\}_{l=1}^L)}\left[\log \frac{\prod_{l=1}^L \prod_{h=1}^{H_l} P(\mathbf{a}_h^l)}{\prod_{l=1}^L \prod_{h=1}^{H_l} Q(\mathbf{a}_h^l)}\right]. \tag{7.4}$$

The logarithm in the first term does not depend on the regression coefficients $\{\boldsymbol{A}^l\}_{l=1}^L$ and neither on $\{\boldsymbol{F}^l\}_{l=1}^{L-1}$. On the other hand, the logarithm on the second term does not depend on $\{\boldsymbol{F}^l\}_{l=1}^L$. Thus,

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{Q(\mathbf{F}^L)}\left[\log \prod_{n=1}^{N} P(y_n \mid \mathbf{f}_{\cdot,n}^L)\right] + \mathbb{E}_{Q(\{\mathbf{A}^l\}_{l=1}^L)}\left[\log \frac{\prod_{l=1}^{L}\prod_{h=1}^{H_l} P(\mathbf{a}_h^l)}{\prod_{l=1}^{L}\prod_{h=1}^{H_l} Q(\mathbf{a}_h^l)}\right] \\ &= \sum_{n=1}^{N} \mathbb{E}_q\big[\log P\big(y_n \mid \mathbf{f}_{\cdot,n}^L\big)\big] - \sum_{l=1}^{L}\sum_{h=1}^{H_l} \mathrm{KL}\big(Q(\mathbf{a}_h^l)\big|P(\mathbf{a}_h^l)\big)\,.\end{aligned} \tag{7.5}$$

DERIVATION OF THE MARGINALS

In this section it is shown that using the assumptions of DVIPs, the n^{th} marginal of the final layer depends only on the n^{th} marginals of the other layers. The variational distribution $Q(\{\boldsymbol{f}^l\}_{l=1}^L)$ factorizes as the product of Gaussian distributions:

$$\begin{aligned}Q(\{\mathbf{F}^l\}_{l=1}^L) &= \prod_{n=1}^{N}\prod_{l=1}^{L}\prod_{h=1}^{H_L} Q(f_{h,n}^l|\mathbf{f}_{\cdot,n}^{l-1}) \\ &= \prod_{n=1}^{N}\prod_{l=1}^{L}\prod_{h=1}^{H_L} \int P(f_{h,n}^l|\mathbf{f}_{\cdot,n}^{l-1},\mathbf{a}_h^l)Q(\mathbf{a}_h^l)d\mathbf{a}_h^l \\ &= \prod_{n=1}^{N}\prod_{l=1}^{L}\prod_{h=1}^{H_L} \mathcal{N}\left(f_{h,n}^l|\hat{m}_{h,n}^l(\mathbf{f}_{\cdot,n}^{l-1}),\hat{v}_{h,n}^l(\mathbf{f}_{\cdot,n}^{l-1})\right)\,.\end{aligned} \tag{7.6}$$

As a result, the n^{th} marginal of the final layer depends only on the n^{th} marginals of the other layers. That is,

$$Q(f_n^L) = \int \prod_{l=1}^{L}\prod_{h=1}^{H_L} \mathcal{N}\left(f_{h,n}^l|\hat{m}_{h,n}^l(\mathbf{f}_{\cdot,n}^{l-1}),\hat{v}_{h,n}^l(\mathbf{f}_{\cdot,n}^{l-1})\right) d\mathbf{f}_{\cdot,n}^1,\ldots,d\mathbf{f}_{\cdot,n}^{L-1}\,. \tag{7.7}$$

8

CONDUCTED EXPERIMENTS

8.1 EXPERIMENTAL SETTINGS

DVIPs have being evaluated on several tasks, including time series interpolation, regression and classification problems. Unless stated otherwise, the number of linear regression coefficients is set to $S = 20$ and a BNN as the IP prior for each unit for all the layers. These BNNs have two hidden layers of 10 units each with *tanh* as the activation function, as in (Ma *et al.*, 2019). The prior mean and variance of each weight and bias in the BNN have being tuned. As no regularizer is used for the prior parameters, the prior mean and variances are constrained to be the same in a layer of the BNN. This configuration avoids over-fitting and leads to improved results.

Figure 14 shows the obtained predictive distributions and learned prior samples of VIPs using a full unconstrained prior BNN (left) and the considered constraint BNN (right). This approach generates usually smoother prior functions and considerably reduces the number of parameters in the model. However, despite providing better results by avoiding over-fitting, there might be datasets where using the full parameterization of the BNN leads to improved results. For example, in problems where overfitting to the training data does not undermine the results on new testing points.

To speed-up computations, at each layer l the generative function that defines the IP prior is shared across units. That is, the function $g_{\boldsymbol{\theta}_h^l}(\cdot, \boldsymbol{z})$ is the same for every unit h in that layer. As a consequence, the prior IP samples only need to be generated once per layer, as in (Ma *et al.*, 2019). This assumption is similar to that of DGPs, which assume shared GP prior parameters for each layer (Salimbeni & Deisenroth, 2017). Under these assumptions and simplifications, each VIP layer contains 12 tunnable parameters in its prior, which are the mean and variance of each weight and bias of the three layers in its BNN.

To evaluate the performance of DVIPs, the model is compared with VIPs (Ma *et al.*, 2019) and DGPs, closely following Salimbeni & Deisenroth (2017). In DGPs 100 shared inducing points in each layer are considered. The employed optimization algorithm is ADAM (Kingma & Ba, 2015) with learning rate 10^{-3}, as in Ma *et al.* (2019). In DGPs, the learning rate is set to 10^{-2}, as in Salimbeni & Deisenroth (2017).

Following Salimbeni & Deisenroth (2017), unless indicated otherwise, in DVIPs and DGPs the input dimension is used as the layer dimensionality for all the inner layers, *i.e* $H_l = M$, for $l = 1, \ldots, L-1$. In DGPs the kernel employed is RBF with ARD (Rasmussen & Williams,

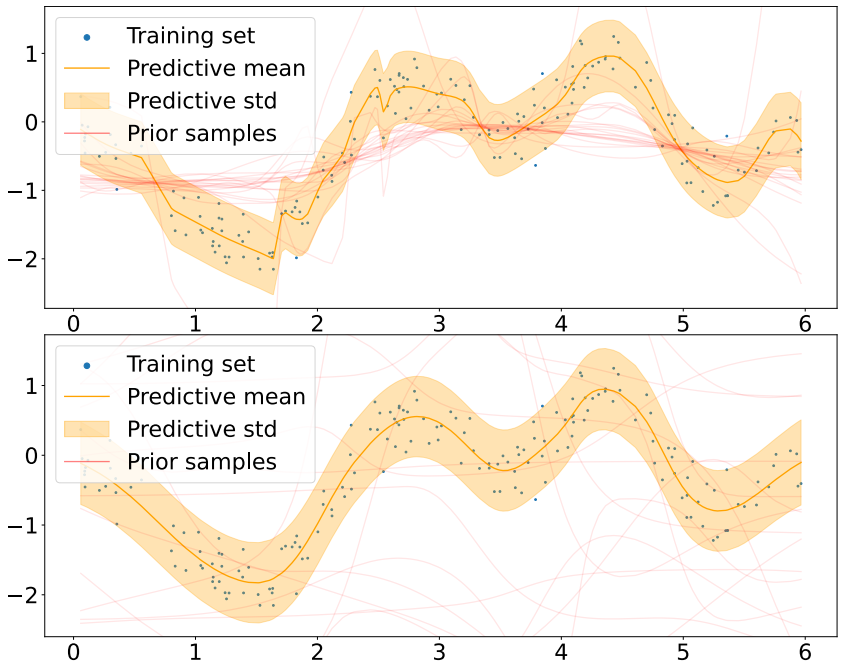


Figure 14: Resulting predictive distribution and prior samples over a toy dataset with full BNN prior (above) and constraint prior BNN (below).

2006). The batch size is 100. All methods are trained for $150,000$ iterations unless indicated otherwise.

The marginal likelihood regularizer described in Ma *et al.* (2019) is not employed in VIPs nor VIP layers since the authors of that paper indicated that they did not use it in practice. Black-box α-divergences are used for VIPs with $\alpha = 0.5$, as suggested in (Ma *et al.*, 2019).

Results are not compared with fBNNs nor FVIs, described in Section 4, because these methods cannot tune the prior IP parameters to the data nor they do consider deep architectures such as the one shown in Figure 13 (right). The used implementation is explained in Section 9.

8.2 RESULTS

Regression UCI benchmarks

DVIPs are compared with DGPs and VIPs on 8 different regression datasets from the UCI Repository (Dua & Graff, 2017). Following common practice, the performance is validated on each dataset using 20 different train-test partitions of the data with 10% test size (Salimbeni & Deisenroth, 2017; Hernández-Lobato & Adams, 2015).

Both DVIPs and DGPs are tested using 2, 3, 4 and 5 layers. Moreover, VIPs, which is equivalent to DVIPs with $L = 1$, is tested with the same prior as DVIPs and using a bigger BNN of 200 units per layer. Furthermore, results are compared with a single sparse GPs, which is equivalent to DGPs for $L = 1$. Figure 15 shows the results obtained in terms of the negative test log-likelihood. Exact results are found in Table 4.

Results show that DVIPs with at least 3 layers performs best on 4 out of the 8 datasets (`Boston`, `Energy`, `Concrete` and `Power`), having comparable results on `Winered` and `Naval` (all methods have zero RMSE on this dataset). DGPs perform best on 2 datasets (`Protein` and `Kin8nm`), but the differences are small. Adding more layers in DVIPs does not lead to over-fitting and it gives similar and often better results (more notably on larger datasets: `Naval`, `Protein`, `Power` and `Kin8nm`). DVIPs also perform better than VIPs most of the times. By contrast, using a more flexible BNN prior in VIPs (*i.e.*, 200 units) does not improve results. Figure 16 shows the training time in seconds of each method. These execution times show that DVIPs are faster than DGPs and faster than VIPs with the 200 units BNN prior. Summing up, DVIPs achieve similar results to those of DGPs, but at a smaller cost.

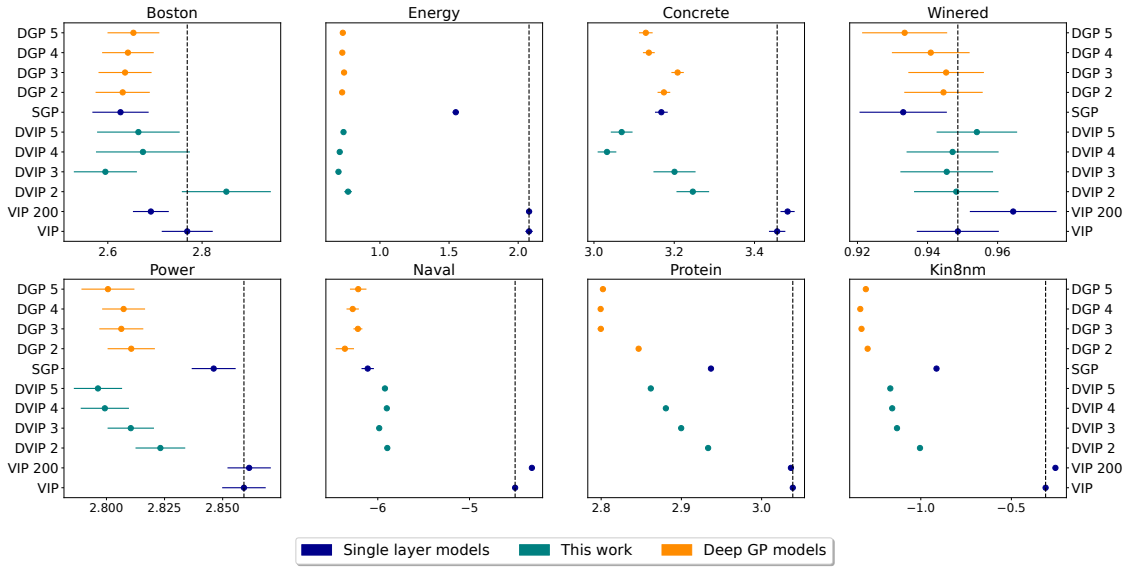


Figure 15: Negative test log-likelihood results on regression UCI benchmark datasets over 20 splits. Standard errors are shown. Lower values (to the left) are better.

Interpolation results

As part of the empirical evaluation, experiments on the CO2 time-series dataset (`https://scrippsco2.ucsd.edu`) are carried out. This dataset consists on CO_2 measurements from the Mauna Loa Observatory, Hawaii, in 1978. The dataset is split in five consecutive and equal parts, with the 2nd and 4th parts used as missing testing data. All models are trained for $100,000$ iterations. Figure 17 shows the predictive distribution of DVIPs and DGPs with $L = 2$ on the data. Results show that DVIPs can capture the data trend in the missing gaps. For DVIPs, samples from the learned prior are shown, which are very smooth. By contrast, a DGPs with RBF kernels fails to capture the data trend, leading to non-smooth priors which produce an over-estimation of the prediction uncertainty. Therefore, the BNN prior considered by DVIPs could be a better choice on this particular dataset. This issue of DGPs can be overcome using compositional kernels (Duvenaud *et al.*, 2014), but that requires to employ kernel search algorithms.

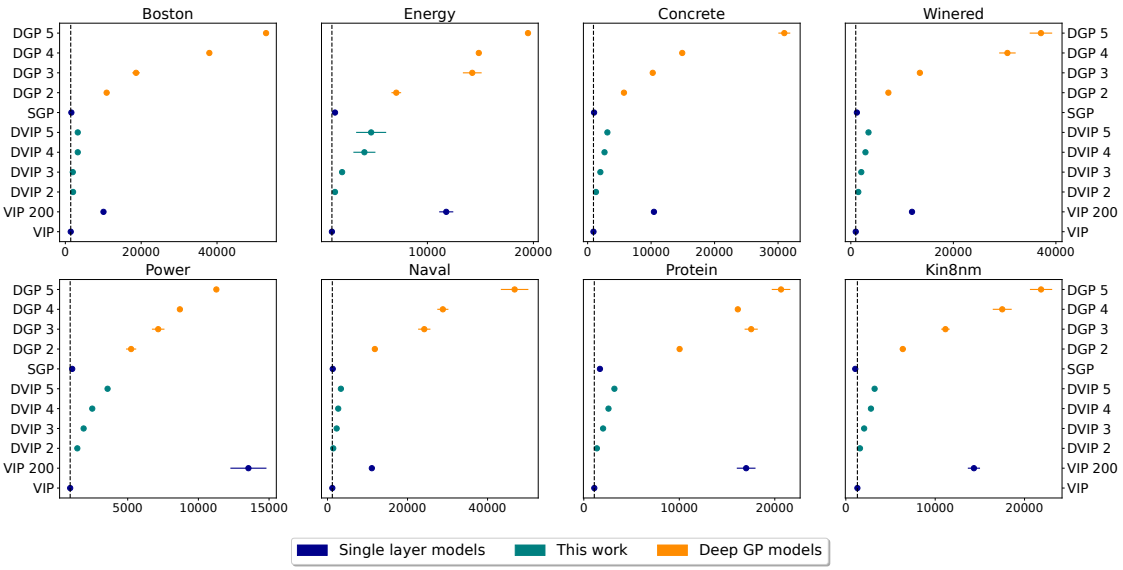


Figure 16: CPU training time (in seconds) on regression UCI benchmark datasets over 20 splits. Standard errors are shown. Lower values (to the left) are better.

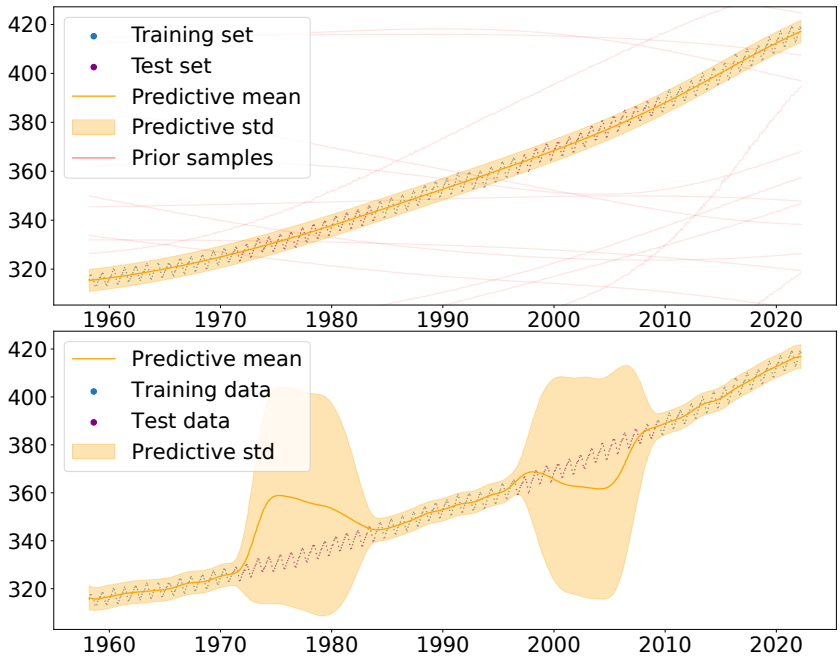


Figure 17: Missing values interpolation results on the `CO2` dataset. Predictive distribution of DVIP (above) and DGP (below) with 2 layers each. Two times the standard deviation is represented.

Large scale regression

Each method is evaluated on 3 large regression datasets. First, the `Year` dataset (UCI) with $515,345$ instances and 90 features, where the original train/test splits are used. Second, the US flight delay (`Airline`) dataset (Dutordoir *et al.*, 2020; Hensman *et al.*, 2017), where following Salimbeni & Deisenroth (2017) the first $700,000$ instances are used for training and

the next $100,000$ for testing. 8 features are considered: *month, day of month, day of week, plane age, air time, distance, arrival time and departure time*. Lastly, data recorded on January, 2015 from the `Taxi` dataset (Salimbeni & Deisenroth, 2017) is used. In this dataset 10 attributes are considered: *time of day, day of week, day of month, month, pickup latitude, pickup longitude, drop-off longitude, drop-off latitude, trip distance and trip duration*. Trips with a duration lower than 10 seconds and larger than 5 hours are removed as in Salimbeni & Deisenroth (2017), leaving $12,695,289$ instances. Results are averaged over 20 train/test splits with 90% and 10% of the data. Here, each method is trained for $500,000$ iterations. The results obtained are shown in Table 1. The last column shows the best result by DGPs, which is achieved for $L = 3$ on each dataset.

It can be observed that DVIPs outperform VIPs on all datasets, and on `Airline` and `Taxi`, the best method are DVIPs. In `Taxi` a SGPs and DGPs give similar results, while DVIPs improve over VIPs. The best method on `Year`, however, are DGPs. Since the difference between DVIPs and DGPs is found in the prior, one could hypothesise that GPs with an RBF kernel may be a better prior on this dataset than the BNN IP prior considered by DVIPs. Therefore, since DVIPs generalize DGPs, DVIPs using GP priors should give similar results to those of DGPs on `Year`.

Table 1: Root Mean Squared Error, Negative Log Likelihood and Continuous Ranked Probability Score results on large scale regression datasets.

RMSE	Single-layer		DVIP				Salimbeni
	SGP	VIP	DVIP 2	DVIP 3	DVIP 4	DVIP 5	Best DGP
Year	9.11	10.23	9.59	9.23	9.24	9.24	**8.88**
Airline	38.57	39.10	37.91	37.82	37.77	**37.71**	37.92
Taxi	554.22 ± 0.32	554.60 ± 0.19	549.28 ± 0.59	$\mathbf{531.42 \pm 1.59}$	547.33 ± 1.03	538.94 ± 2.23	552.90 ± 0.33
NLL	Single-layer		DVIP				Salimbeni
	SGP	VIP	DVIP 2	DVIP 3	DVIP 4	DVIP 5	Best DGP
Year	3.628	3.744	3.679	3.639	3.639	3.639	**3.599**
Airline	5.095	5.110	5.080	5.072	5.076	**5.070**	5.074
Taxi	7.73 ± 0.00	7.73 ± 0.00	7.72 ± 0.00	$\mathbf{7.69 \pm 0.00}$	7.72 ± 0.00	7.70 ± 0.00	7.73 ± 0.00
CRPS	Single-layer		DVIP				Salimbeni
	SGP	VIP	DVIP 2	DVIP 3	DVIP 4	DVIP 5	Best DGP
Year	4.826	5.457	5.096	4.885	4.912	4.871	**4.681**
Airline	18.12	18.65	18.00	17.77	17.75	17.65	**17.51**
Taxi	283.79 ± 0.19	284.22 ± 0.20	282.09 ± 0.32	$\mathbf{274.65 \pm 0.68}$	281.28 ± 0.44	277.60 ± 0.90	282.99 ± 0.21

Image classification

Binary classification experiments were carried out using the `Rectangles` dataset (Salimbeni & Deisenroth, 2017) and the multi-class dataset `MNIST` (Deng, 2012). Each dataset has 28×28 pixels images. The `Rectangles` dataset has $12,000$ images of a (non-square) rectangle. The task is to determine if the height is larger than the width. For this dataset, a probit likelihood is employed for all the models. The `MNIST` dataset has $70,000$ images of handwritten digits. The labels correspond with each digit. In this case, the robust-max multi-class likelihood (Hernández-Lobato *et al.*, 2011) is used. Given the size of `Rectangles`, $20,000$ iterations are

enough to ensure convergence. Results on this dataset are averaged over 10 different random seeds, but the standard errors are omitted as they are always lower than 0.1. The original train-test splits are used for both dataset. Examples of images from these datasets are shown in Figure 18.

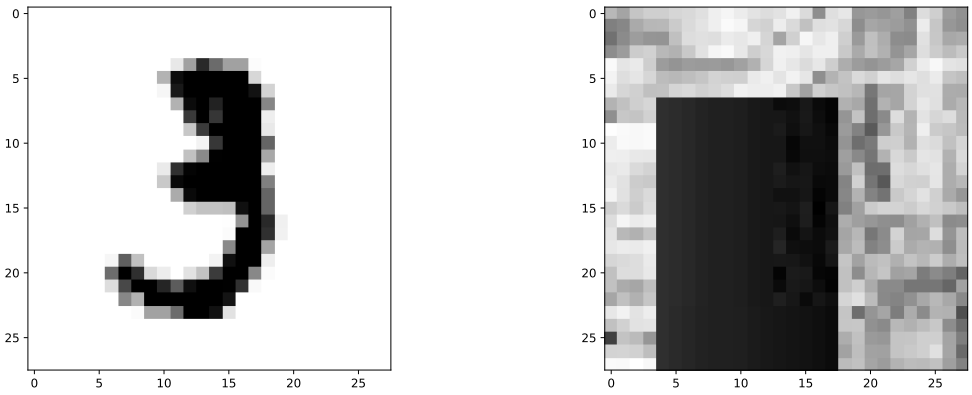


Figure 18: Examples of the image classification datasets: MNIST (left) and Rectangles (right). The MNIST digit corresponds to a 3 and the Rectangles image to one where the height of the inner image is higher than its width.

Critically, DVIP's capability to use more flexible priors is exploited. Specifically, in the first layer a convolutional NN (CNN) prior with two layers of 4 and 8 channels respectively is employed. The inner dimensions of DVIPs and DGPs are fixed to 30, as in Salimbeni & Deisenroth (2017). No input propagation is used in the first layer. The results obtained are shown in Table 2.

Results show that DVIPs obtain much better results than those of DGPs and VIPs. In particular, DVIPs increase the accuracy by 11% on `Rectangles` compared to DGPs, probably as a consequence of the CNN prior considered in the first layer of the network being more suited for image-based datasets.

Table 2: Results on image classification datasets.

MNIST	Single-layer		DVIP		Salimbeni	
	SGP	VIP	DVIP 2	DVIP 3	DGP 2	DGP 3
Accuracy (%)	96.33	98.14	**98.57**	98.38	97.85	97.97
Likelihood	−0.138	−0.133	−0.075	−0.081	−0.081	**−0.073**

Rectangles	Single-layer		DVIP				Salimbeni
	SGP	VIP	DVIP 2	DVIP 3	DVIP 4	DVIP 5	DGP 3
Accuracy (%)	73.64	85.50	87.92	**88.40**	87.89	86.82	77.18
Likelihood	−0.526	−0.349	−0.294	**−0.280**	−0.286	−0.301	−0.472
AUC	0.826	0.931	0.952	**0.956**	0.952	0.945	0.857

Large scale classification

As a final set of experiments, DVIP is evaluated on two massive binary datasets; `SUSY` and `HIGGS`, with 5.5 million and 10 million instances, respectively. These datasets contain Monte Carlo physics simulations to detect the presence of the Higgs boson and super-symmetry (Baldi *et al.*, 2014). The original train-test splits of the data are used and $500,000$ iterations for training. The AUC metric is reported for comparison with Baldi *et al.* (2014) and Salimbeni & Deisenroth (2017). Results are shown in Table 3. It can be observed that DVIPs achieve the highest performance on `SUSY` (AUC of 0.8756) which is comparable to that of DGPs (0.8751) and to the best reported results in Baldi *et al.* (2014). Namely, shallow NNs (NN, 0.875), deep NN (DNN, 0.876) and boosted decision trees (BDT, 0.863). On `HIGGS`, despite seeing an steady improvement over VIPs by using additional layers, the performance is worse than that of DGPs (AUC 0.8325) and worse than the best model reported in Baldi *et al.* (2014), a 5 layer DNN (AUC 0.885). Similarly to the `Year` dataset, GPs with an RBF kernel may be a better prior here, and that DVIPs using GP priors should give similar results to those of DGPs.

Table 3: Results on large classification datasets.

SUSY	Single-layer		DVIP				Salimbeni
	SGP	VIP	DVIP 2	DVIP 3	DVIP 4	DVIP 5	DGP 4
Accuracy (%)	79.76	78.68	80.13	80.14	**80.22**	**80.22**	80.06
Likelihood	-0.436	-0.4569	-0.4299	-0.4290	-0.4279	$\mathbf{-0.4278}$	-0.432
AUC	0.8727	0.8572	0.8742	0.8749	0.8755	**0.8756**	0.8751
HIGGS	SGP	VIP	DVIP 2	DVIP 3	DVIP 4	DVIP 5	DGP 5
Accuracy (%)	69.94	57.46	66.10	69.86	70.42	72.02	**74.91**
Likelihood	-0.5750	-0.6715	-0.6143	-0.5731	-0.5670	-0.5453	$\mathbf{-0.500}$
AUC	0.7693	0.6247	0.7196	0.7704	0.7782	0.7962	**0.8324**

Table 4: Negative Log Likelihood, Root Mean Squared Error and Continuous Ranked Probability Score results on regression UCI benchmark datasets.

NLL	Single-layer			DVIP				Salimbeni			
	SGP	VIP	VIP 200	DVIP 2	DVIP 3	DVIP 4	DVIP 5	DGP 2	DGP 3	DGP 4	DGP 5
Boston	2.62 ± 0.05	2.76 ± 0.05	2.69 ± 0.03	2.85 ± 0.09	$\mathbf{2.59 \pm 0.06}$	2.67 ± 0.09	2.66 ± 0.08	2.63 ± 0.05	2.63 ± 0.05	2.64 ± 0.05	2.65 ± 0.05
Energy	1.54 ± 0.02	2.07 ± 0.02	2.07 ± 0.02	0.76 ± 0.02	$\mathbf{0.70 \pm 0.01}$	$\mathbf{0.70 \pm 0.01}$	0.73 ± 0.01	0.72 ± 0.01	0.74 ± 0.01	0.72 ± 0.01	0.73 ± 0.01
Concrete	3.16 ± 0.01	3.45 ± 0.02	3.48 ± 0.01	3.24 ± 0.04	3.20 ± 0.05	$\mathbf{3.03 \pm 0.02}$	3.06 ± 0.02	3.17 ± 0.01	3.20 ± 0.01	3.13 ± 0.01	3.12 ± 0.01
Winered	$\mathbf{0.93 \pm 0.01}$	0.94 ± 0.01	0.96 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	0.95 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	$\mathbf{0.93 \pm 0.01}$
Power	2.84 ± 0.00	2.85 ± 0.00	2.86 ± 0.00	2.82 ± 0.01	2.81 ± 0.00	$\mathbf{2.79 \pm 0.01}$	$\mathbf{2.79 \pm 0.01}$	2.81 ± 0.01	2.80 ± 0.00	2.80 ± 0.00	2.80 ± 0.01
Protein	2.93 ± 0.00	3.03 ± 0.00	3.03 ± 0.00	2.93 ± 0.00	2.89 ± 0.00	2.88 ± 0.00	2.86 ± 0.00	2.84 ± 0.00	$\mathbf{2.79 \pm 0.00}$	$\mathbf{2.79 \pm 0.00}$	2.80 ± 0.00
Naval	-6.11 ± 0.06	-4.50 ± 0.02	-4.31 ± 0.00	-5.89 ± 0.02	-5.98 ± 0.01	-5.90 ± 0.01	-5.92 ± 0.01	$\mathbf{-6.35 \pm 0.09}$	-6.21 ± 0.04	-6.27 ± 0.06	-6.21 ± 0.08
Kin8nm	-0.91 ± 0.00	-0.31 ± 0.00	-0.25 ± 0.00	-1.00 ± 0.00	-1.13 ± 0.00	-1.15 ± 0.00	-1.16 ± 0.00	-1.29 ± 0.00	-1.32 ± 0.00	$\mathbf{-1.33 \pm 0.00}$	1.30 ± 0.00

RMSE	Single-layer			DVIP				Salimbeni			
	SGP	VIP	VIP 200	DVIP 2	DVIP 3	DVIP 4	DVIP 5	DGP 2	DGP 3	DGP 4	DGP 5
Boston	$\mathbf{3.48 \pm 0.17}$	4.78 ± 0.28	4.49 ± 0.28	3.87 ± 0.19	3.50 ± 0.20	3.60 ± 0.19	3.66 ± 0.21	3.51 ± 0.18	3.53 ± 0.19	3.55 ± 0.20	3.56 ± 0.20
Energy	1.07 ± 0.03	2.57 ± 0.08	2.68 ± 0.07	0.52 ± 0.01	0.47 ± 0.01	$\mathbf{0.46 \pm 0.01}$	0.47 ± 0.01	$\mathbf{0.46 \pm 0.01}$	0.47 ± 0.01	$\mathbf{0.46 \pm 0.01}$	$\mathbf{0.46 \pm 0.01}$
Concrete	5.84 ± 0.12	7.75 ± 0.15	8.06 ± 0.16	6.01 ± 0.16	5.68 ± 0.18	$\mathbf{5.13 \pm 0.12}$	5.27 ± 0.13	5.86 ± 0.12	6.01 ± 0.12	5.54 ± 0.11	5.52 ± 0.12
Winered	$\mathbf{0.61 \pm 0.00}$	0.62 ± 0.00	0.63 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00	0.62 ± 0.00
Power	4.15 ± 0.03	4.21 ± 0.03	4.22 ± 0.03	4.06 ± 0.04	4.01 ± 0.04	3.97 ± 0.04	$\mathbf{3.95 \pm 0.04}$	4.00 ± 0.04	3.98 ± 0.03	3.99 ± 0.03	3.96 ± 0.04
Protein	4.56 ± 0.01	5.05 ± 0.01	5.04 ± 0.01	4.54 ± 0.01	4.40 ± 0.01	4.33 ± 0.01	4.26 ± 0.01	4.17 ± 0.01	$\mathbf{4.00 \pm 0.01}$	4.01 ± 0.01	4.02 ± 0.01
Naval	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
Kin8nm	0.09 ± 0.00	0.17 ± 0.00	0.18 ± 0.00	0.08 ± 0.00	0.07 ± 0.00	0.07 ± 0.00	0.07 ± 0.00	$\mathbf{0.06 \pm 0.00}$	$\mathbf{0.06 \pm 0.00}$	$\mathbf{0.06 \pm 0.00}$	$\mathbf{0.06 \pm 0.00}$

CRPS	Single-layer			DVIP				Salimbeni			
	SGP	VIP	VIP 200	DVIP 2	DVIP 3	DVIP 4	DVIP 5	DGP 2	DGP 3	DGP 4	DGP 5
Boston	1.79 ± 0.05	2.25 ± 0.08	2.13 ± 0.08	$1.91 \pm .06$	$\mathbf{1.76 \pm 0.07}$	1.81 ± 0.07	1.78 ± 0.06	1.79 ± 0.05	1.80 ± 0.06	1.80 ± 0.06	1.81 ± 0.06
Energy	0.62 ± 0.01	1.27 ± 0.04	1.30 ± 0.03	0.28 ± 0.00	$\mathbf{0.26 \pm 0.00}$	$\mathbf{0.26 \pm 0.00}$	$\mathbf{0.26 \pm 0.00}$	$\mathbf{0.26 \pm 0.00}$	$\mathbf{0.26 \pm 0.00}$	$\mathbf{0.26 \pm 0.00}$	$\mathbf{0.26 \pm 0.00}$
Concrete	3.20 ± 0.05	4.29 ± 0.08	4.43 ± 0.08	3.26 ± 0.07	3.03 ± 0.09	$\mathbf{2.74 \pm 0.05}$	2.83 ± 0.05	3.21 ± 0.05	3.31 ± 0.05	3.05 ± 0.05	3.04 ± 0.05
Winered	$\mathbf{0.34 \pm 0.00}$	$\mathbf{0.34 \pm 0.00}$	0.35 ± 0.00	$\mathbf{0.34 \pm 0.00}$	$\mathbf{0.34 \pm 0.00}$	$\mathbf{0.34 \pm 0.00}$	$\mathbf{0.34 \pm 0.00}$	$\mathbf{0.34 \pm 0.00}$	$\mathbf{0.34 \pm 0.00}$	$\mathbf{0.34 \pm 0.00}$	$\mathbf{0.34 \pm 0.00}$
Power	2.27 ± 0.01	2.31 ± 0.01	2.31 ± 0.01	2.21 ± 0.01	2.18 ± 0.01	$\mathbf{2.14 \pm 0.01}$	$\mathbf{2.14 \pm 0.01}$	2.17 ± 0.01	2.16 ± 0.01	2.17 ± 0.01	2.15 ± 0.01
Protein	2.56 ± 0.00	2.87 ± 0.00	2.86 ± 0.01	2.54 ± 0.00	2.43 ± 0.00	2.38 ± 0.00	2.33 ± 0.00	2.31 ± 0.00	$\mathbf{2.19 \pm 0.00}$	$\mathbf{2.19 \pm 0.00}$	2.20 ± 0.00
Naval	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
Kin8nm	0.05 ± 0.00	0.09 ± 0.0	0.10 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	0.04 ± 0.00	$\mathbf{0.03 \pm 0.00}$	$\mathbf{0.03 \pm 0.00}$	$\mathbf{0.03 \pm 0.00}$	$\mathbf{0.03 \pm 0.00}$

9

IMPLEMENTATION DETAILS

The implementation of DVIPs is done using the open source machine learning framework `PyTorch` and publicly available in the Github Repository DeepVIP. The code is structured in different folders containing either Python scripts to run different experiments or Python files defining the model and making the usage more user-friendly.

More precisely, the code is divided in the following folders:

- `src`: Contains the main definition classes of DVIPs. Containing its layers, random samplers, prior distributions and likelihoods implementation.
- `scripts`: Contains the main scripts used for running the experiments.
- `utils`: Contains the auxiliary functions and classes the make simplify the usage of DVIPs. For example, the specification of `argparser` to run the experiments using flags on the python call.

The main provided way of using the model is to use one of the scripts. For example, experiments with different splits are made using the `split.py` scripts. In order to specify the experiment parameters, these must be passed as parameters to the python call, e.g,

```
python scripts/split.py --dataset boston  --iterations 150000  --split 0
                        --bnn_layer SimplerBayesLinear --vip_layers 2
```

can be used to run a experiment on the first split of Boston dataset, with $150,000$ iterations, 2 layers and the simplified BNN. All the available FLAGS can be seeing using `--help`. Experiments without splits are performed using this same script but the corresponding flag serves no purpose. Moreover, the CO2 experiments are done using the `missing_gaps.py` script:

```
python scripts/missing_gaps.py --dataset CO2 --iterations 100000 --vip_layers 2
                               --bnn_layer SimplerBayesLinear
```

The available datasets are specified in `datasets.py` inside `utils`. Most of them are retrieved directly from urls in the UCI website. However, some of them, as the Rectangles one, must be located locally inside a folder called `data`. This precise dataset can be downloaded from the following online resource: http://www.iro.umontreal.ca/~lisa/icml2007data/.

CONCLUSIONS AND FURTHER WORK

Deep Variational Implicit Process (DVIPs) are defined as a model based on the concatenation of implicit processes (IPs) as the prior for the latent function. The conducted experiments demonstrate that DVIPs can be used on a variety of regression and classification problems with no need of hand-tuning. Results show that DVIPs surpass or match the performance of single layer VIPs and GPs. It also gives similar and sometimes better results than those of deep GPs (DGPs). However, the main comparative advantage of DVIPs is that is has less computational cost than DGPs. Experiments have also demonstrated that DVIPs are both effective and scalable on a wide range of tasks. DVIPs do not seem to over-fit on small datasets by increasing the depth, and on large datasets, extra layers often improve performance. It is also shown that increasing the number of layers is far more effective than increasing the complexity of the prior of single-layer VIPs, which heavily affects the training time of the model. Besides from the added computation time, which is rather minor, no drawbacks to the use of DVIPs instead of single-layer VIPs are seen, but rather significant benefits.

The use of domain specific priors, e.g. considering of CNNs in the first layer, has demonstrated to give outstanding results in image-based datasets compared to other GP methods. This establishes a new use of IPs with not-so-general prior functions. The employment of these priors on other domain specific tasks, such as forecasting or data encoding, is foreseen as an emerging field of study. The prior flexibility also results in a generalization of DGPs. As a matter of fact, DVIPs should give similar results to those of DGPs if a GP is considered as the IP prior for each unit.

Despite the good results, DVIPs presents some limitations: first of all, the implicit prior works as a black-box from the interpretability point of view. The prior parameters do not represent a clear property of the model in comparison to the original GP's kernel parameters, such as the length and amplitude. Furthermore, even though using 20 samples from the prior has shown to give remarkable results in most cases, there might be situations where this number must be increased, having a big impact in the model's training time.

Future work could include using more flexible base models in DVIPs. For example, as an alternative to the Bayesian linear approximation, SIP (Santana *et al.*, 2021) could be considered instead, which uses an sparse approach to approximate the posterior. The use of domain-specific prior functions in DVIPs, such as LSTM for time series or text processing, could also be interesting. Finally, using DVIPs in the context of autoencoders, generalizing DGPs on this subject, is another line of future research (Domingues *et al.*, 2018).

ACKNOWLEDGMENTS

I would first like to thank my tutors, Daniel Hernández Lobato and Simón Rodríguez Santana, who have helped me greatly with their valuable guidance, corrections and advice in writing this document. I want to thank you for your patient support answering all my questions and doubts over the course of this project.

Also, I gratefully acknowledge the use of the facilities of Centro de Computación Científica (CCC) for providing the necessary means to conduct the experiments shown in this work.

BIBLIOGRAPHY

Anjos, Ofélia, Iglesias, Carla, Peres, Fátima, Martínez, Javier, Garcia, Angela, & Taboada, Javier. 2015. Neural networks applied to discriminate botanical origin of honeys. *Food chemistry*, **175**, 128–136.

Baldi, Pierre, Sadowski, Peter, & Whiteson, Daniel. 2014. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, **5**(1), 1–9.

Balog, Matej, Lakshminarayanan, Balaji, Ghahramani, Zoubin, Roy, Daniel M, & Teh, Yee Whye. 2016. The mondrian kernel. *arXiv preprint arXiv:1606.05241*.

Barber, David. 2012. *Bayesian reasoning and machine learning*. Cambridge University Press.

Bergmann, Sören, Stelzer, Sören, & Strassburger, Steffen. 2014. On the use of artificial neural networks in simulation-based manufacturing control. *Journal of Simulation*, **8**(1), 76–90.

Bonilla, Edwin V., Krauth, Karl, & Dezfouli, Amir. 2018. *Generic Inference in Latent Gaussian Process Models*.

Bui, Thang, Hernández-Lobato, Daniel, Hernandez-Lobato, Jose, Li, Yingzhen, & Turner, Richard. 2016. Deep Gaussian processes for regression using approximate expectation propagation. *Pages 1472–1481 of: International conference on machine learning*.

Cho, Youngmin, & Saul, Lawrence. 2009. Kernel methods for deep learning. *Advances in neural information processing systems*, **22**.

Cutajar, K., Bonilla, E. V., Michiardi, P., & Filippone, M. 2017. Random Feature Expansions for Deep Gaussian Processes. *Pages 884–893 of: International Conference on Machine Learning*.

Damianou, Andreas, & Lawrence, Neil D. 2013. Deep Gaussian processes. *Pages 207–215 of: Artificial intelligence and statistics*.

Deng, Li. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, **29**, 141–142.

Domingues, Rémi, Michiardi, Pietro, Zouaoui, Jihane, & Filippone, Maurizio. 2018. Deep Gaussian Process autoencoders for novelty detection. *Machine Learning*, **107**(8), 1363–1383.

Dua, Dheeru, & Graff, Casey. 2017. *UCI Machine Learning Repository*.

Dutordoir, Vincent, Durrande, Nicolas, & Hensman, James. 2020. Sparse Gaussian processes with spherical harmonic features. *Pages 2793–2802 of: International Conference on Machine Learning*.

Duvenaud, David, Rippel, Oren, Adams, Ryan, & Ghahramani, Zoubin. 2014. Avoiding pathologies in very deep networks. *Pages 202–210 of: Artificial Intelligence and Statistics*.

Flam-Shepherd, Daniel, Requeima, James, & Duvenaud, David. 2017. Mapping Gaussian process priors to Bayesian neural networks. *In: NIPS Bayesian deep learning workshop*, vol. 3.

Gal, Yarin. 2016. *Uncertainty in Deep Learning*. Ph.D. thesis, University of Cambridge.

Gal, Yarin, & Turner, Richard. 2015. Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. *Pages 655–664 of: International Conference on Machine Learning*. PMLR.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. 2013. *Bayesian data analysis*. CRC press.

Ghahramani, Zoubin. 2015. Probabilistic machine learning and artificial intelligence. *Nature*, **521**(7553), 452–459.

Hajjo, Rima, Sabbah, Dima A, Bardaweel, Sanaa K, & Tropsha, Alexander. 2021. Identification of tumor-specific MRI biomarkers using machine learning (ML). *Diagnostics*, **11**(5), 742.

Havasi, Marton, Hernández-Lobato, José Miguel, & Murillo-Fuentes, Juan José. 2018. Inference in Deep Gaussian Processes using Stochastic Gradient Hamiltonian Monte Carlo. *In: Advances in Neural Information Processing Systems*, vol. 31.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian. 2016. Identity mappings in deep residual networks. *Pages 630–645 of: European conference on computer vision*.

Hensman, James, Fusi, Nicolò, & Lawrence, Neil D. 2013. Gaussian Processes for Big Data. *Page 282–290 of: Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*.

Hensman, James, de G. Matthews, Alexander G., Filippone, Maurizio, & Ghahramani, Zoubin. 2015. *MCMC for Variationally Sparse Gaussian Processes*.

Hensman, James, Durrande, Nicolas, Solin, Arno, *et al.* 2017. Variational Fourier Features for Gaussian Processes. *Journal of Machine Learning Research*, **18**(1), 5537–5588.

Hernández-Lobato, Daniel, Hernández-Lobato, Jose, & Dupont, Pierre. 2011. Robust Multi-Class Gaussian Process Classification. *In: Advances in Neural Information Processing Systems*, vol. 24.

Hernandez-Lobato, Jose, Li, Yingzhen, Rowland, Mark, Bui, Thang, Hernández-Lobato, Daniel, & Turner, Richard. 2016. Black-box alpha divergence minimization. *Pages 1511–1520 of: International Conference on Machine Learning*.

Hernández-Lobato, José Miguel, & Adams, Ryan. 2015. Probabilistic backpropagation for scalable learning of Bayesian neural networks. *Pages 1861–1869 of: International conference on machine learning*.

Hornik, Kurt, Stinchcombe, Maxwell, & White, Halbert. 1989. Multilayer feedforward networks are universal approximators. *Neural networks*, **2**(5), 359–366.

Jordan, Michael I, Ghahramani, Zoubin, Jaakkola, Tommi S, & Saul, Lawrence K. 1999. An introduction to variational methods for graphical models. *Machine learning*, **37**(2), 183–233.

Kalchbrenner, Nal, & Blunsom, Phil. 2013. Recurrent continuous translation models. *Pages 1700–1709 of: Proceedings of the 2013 conference on empirical methods in natural language processing*.

Kingma, D. P., & Ba, J. 2015. ADAM: a method for stochastic optimization. *Pages 1–15 of: Intrernational Conference on Learning Representations*.

Kingma, D. P., & Welling, M. 2014. Auto-Encoding Variational Bayes. *In: International Conference on Learning Representations*.

Lawrence, Neil D, & Moore, Andrew J. 2007. Hierarchical Gaussian process latent variable models. *Pages 481–488 of: Proceedings of the 24th international conference on Machine learning*.

Lee, Jaehoon, Bahri, Yasaman, Novak, Roman, Schoenholz, Samuel S., Pennington, Jeffrey, & Sohl-Dickstein, Jascha. 2018. Deep Neural Networks as Gaussian Processes. *In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Li, Yingzhen, & Gal, Yarin. 2017. Dropout inference in Bayesian neural networks with alpha-divergences. *Pages 2052–2061 of: International conference on machine learning*.

Li, Yingzhen, & Turner, Richard E. 2016. Rényi Divergence Variational Inference. *In: Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc.

Li, Yingzhen, Hernández-Lobato, José Miguel, & Turner, Richard E. 2015. Stochastic expectation propagation. *Advances in neural information processing systems*, **28**.

Ma, Chao, & Hernández-Lobato, José Miguel. 2021. Functional Variational Inference based on Stochastic Process Generators. *Pages 21795–21807 of: Advances in Neural Information Processing Systems*, vol. 34.

Ma, Chao, Li, Yingzhen, & Hernández-Lobato, José Miguel. 2019. Variational implicit processes. *Pages 4222–4233 of: International Conference on Machine Learning*.

MacKay, David JC. 1995. Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks. *Network: computation in neural systems*, **6**(3), 469.

Mercer, J. 1909. Functions ofpositive and negativetypeand theircommection with the theory ofintegral equations. *Philos. Trinsdictions Rogyal Soc*, **209**, 4–415.

Minka, Thomas P. 2013. Expectation propagation for approximate Bayesian inference. *arXiv preprint arXiv:1301.2294*.

Minka, Tom. 2005. *Divergence Measures and Message Passing*. Tech. rept. MSR-TR-2005-173.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, & Riedmiller, Martin. 2013. Playing Atari With Deep Reinforcement Learning. *In: NIPS Deep Learning Workshop*.

Murphy, Kevin P. 2012. *Machine learning: a probabilistic perspective*. MIT press.

Neal, Radford M. 1996. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media.

Quiñonero-Candela, Joaquin, & Rasmussen, Carl Edward. 2005. A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, **6**, 1939–1959.

Rahimi, Ali, & Recht, Benjamin. 2007. Random Features for Large-Scale Kernel Machines. *Pages 1177–1184 of: Advances in Neural Information Processing Systems*.

Rasmussen, C. E., & Williams, C. K. I. 2006. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Rasmussen, Carl Edward. 2003. Gaussian processes in machine learning. *Pages 63–71 of: Summer school on machine learning*.

Rényi, Alfréd. 1961. On measures of entropy and information. *Pages 547–562 of: Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, vol. 4. University of California Press.

Salimbeni, Hugh, & Deisenroth, Marc. 2017. Doubly Stochastic Variational Inference for Deep Gaussian Processes. *In: Advances in Neural Information Processing Systems*, vol. 30.

Santana, Simón Rodríguez, & Hernández-Lobato, Daniel. 2022. Adversarial α-divergence minimization for Bayesian approximate inference. *Neurocomputing*, **471**, 260–274.

Santana, Simón Rodríguez, Zaldivar, Bryan, & Hernández-Lobato, Daniel. 2021. Sparse Implicit Processes for Approximate Inference. *arXiv preprint arXiv:2110.07618*.

Shawe-Taylor, John, Cristianini, Nello, *et al.* 2004. *Kernel methods for pattern analysis*. Cambridge university press.

Shi, Jiaxin, Sun, Shengyang, & Zhu, Jun. 2018. A spectral approach to gradient estimation for implicit distributions. *Pages 4644–4653 of: International Conference on Machine Learning*.

Snelson, Edward, & Ghahramani, Zoubin. 2006. Sparse Gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, **18**, 1257.

Sun, Shengyang, Zhang, Guodong, Shi, Jiaxin, & Grosse, Roger. 2019. Functional variational Bayesian neural networks. *In: International Conference on Learning Representations*.

Titsias, Michalis. 2009. Variational learning of inducing variables in sparse Gaussian processes. *Pages 567–574 of: Artificial intelligence and statistics*. PMLR.

Williams, Christopher K, & Rasmussen, Carl Edward. 2006. *Gaussian processes for machine learning*. Vol. 2. MIT press Cambridge, MA.

Yu, Haibin, Chen, Yizhou, Low, Bryan Kian Hsiang, Jaillet, Patrick, & Dai, Zhongxiang. 2019. Implicit posterior variational inference for deep Gaussian processes. *Advances in Neural Information Processing Systems*, **32**, 14475–14486.

Zhu, Huaiyu, & Rohwer, Richard. 1995. Information geometric measurements of generalisation.