A Tutorial on Sparse Gaussian Processes and Variational Inference

Felix Leibfried[1], Vincent Dutordoir[1,2], ST John[1], and Nicolas Durrande[1]

[1]Secondmind, Cambridge (UK)
[2]University of Cambridge, Cambridge (UK)

Abstract: Gaussian processes (GPs) provide a mathematically elegant framework for Bayesian inference and they can offer principled uncertainty estimates for a large range of problems. For example, if we consider certain regression problems with Gaussian likelihoods, a GP model enjoys a posterior in closed form. However, identifying the posterior GP scales cubically with the number of training examples and furthermore requires to store all training examples in memory. In order to overcome these practical obstacles, sparse GPs have been proposed that approximate the true posterior GP with a set of pseudo-training examples (a.k.a. inducing inputs or inducing points). Importantly, the number of pseudo-training examples is user-defined and enables control over computational and memory complexity. In the general case, sparse GPs do not enjoy closed-form solutions and one has to resort to approximate inference. In this context, a convenient choice for approximate inference is variational inference (VI), where the problem of Bayesian inference is cast as an optimization problem—namely, to maximize a lower bound of the logarithm of the marginal likelihood. This paves the way for a powerful and versatile framework, where pseudo-training examples are treated as optimization arguments of the approximate posterior that are jointly identified together with hyperparameters of the generative model (i.e. prior and likelihood) in the course of training. The framework can naturally handle a wide scope of supervised learning problems, ranging from regression with heteroscedastic and non-Gaussian likelihoods to classification problems with discrete labels, but also multilabel problems (where the regression or classification targets are multidimensional). The purpose of this tutorial is to provide access to the basic matter for readers without prior knowledge in both GPs and VI. It turns out that a proper exposition to the subject enables also convenient access to more recent advances in the field of GPs (like importance-weighted VI as well as interdomain, multioutput and deep GPs) that can serve as an inspiration for exploring new research ideas.

Contents

1 Introduction

Gaussian Processes (GPs) [Rasmussen and Williams, 2006] are a natural way to generalize the concept of a multivariate normal distribution. While a multivariate normal distribution describes random variables that are vectors, a GP describes random variables that are real-valued functions defined over some input domain. Imagine the input domain are the real numbers, then the random variable described by a GP can be thought of as a "vector" of uncountably infinite extent and with infinite resolution that is "indexed" by a real number rather than a discrete index. GPs allow however for a wider range of input domains such as Euclidean vector spaces, but also non-continuous input domains like sets containing graph-theoretical objects or character sequences, and many more.

GPs are a popular tool for regression where the goal is to identify an unknown real-valued function given noisy function observations at some input locations. More precisely, given N input/output tuples $\{(X_n, y_n)\}_{n=1}^N$ where y_n is a scalar and X_n an input from some input domain, the modelling assumption is that the data has been generated by $y_n = f(X_n) + \varepsilon_n$ where $f(\cdot)$ is a real-valued function with input X_n that is sampled from a GP, and where ε_n are scalar i.i.d. random variables corresponding to observation noise. In this context, the prior knowledge on the data generation process can be encapsulated in the distribution of $f(\cdot)$, and the likelihood $p(y_n|f(X_n))$ (i.e. the observation model) determines how likely a noisy function observation y_n is given the corresponding noise-free function observation $f(X_n)$. If the observation noise is Gaussian, the posterior process is also a GP that enjoys a closed-form expression. In order to compute the posterior GP, one has to resort to Bayes' rule to infer the posterior distribution over functions given the input/output tuples and the likelihood. It turns out that the denominator in Bayes' rule, known as the marginal likelihood that depends on both the prior GP and the likelihood, provides a natural way to identify point estimates for hyperparameters of the generative model that are not subject to inference [Bishop, 2006].

In the case of general regression or classification problems, the exact posterior process is usually no longer a GP. In logistic regression for example, the noisy function observations are binary values $y_n \in \{0, 1\}$ (which is why logistic regression is actually a classification problem). The likelihood is a Bernoulli distribution whose mean is obtained by squashing the output of a real-valued function $f(X_n)$ evaluated at the input X_n (represented by a GP) through a sigmoid function $\sigma(f(X_n))$. This yields a probability value $\in [0, 1]$ for each input location indicating the probability of the observed noisy function value y_n being 1. But even regression problems that have Gaussian likelihoods are not unproblematic. It turns out that computing the exact posterior GP requires to store and invert an $(N \times N)$-matrix that is quadratic in the amount of training data N. This means quadratic memory consumption and cubic computational complexity—both of which are infeasible for large data sets.

All of these problems can be addressed with recent advances in the field of GP research: sparse GPs [Titsias, 2009]. Sparse GPs limit the amount of pseudo data that is used to represent a (possibly non-Gaussian) posterior process, where the limit is user-defined and determines memory and computational complexity. Intuitively, in a regression problem that has a closed-form solution, the optimal sparse GP should be "as close as possible" to the true intractable posterior GP. In this regard, "as close as possible" can be for example defined as low Kullback-Leibler (KL) divergence between the sparse GP and the true posterior GP, and the goal is to identify the sparse GP's pseudo data in such a way that this KL becomes minimal. In general, it is not possible to identify such optimal sparse GPs in closed-form solution. One way to approximate optimal sparse GPs is to resort to optimization and gradient-based methods, one particular example of which is variational inference (VI) that is equivalent to minimizing said KL divergence. Other examples comprise Markov-chain-Monte-Carlo methods and expectation propagation [Hensman et al., 2015a, Bui et al., 2017]. However, in this tutorial, we put emphasis on VI due to its popularity and convenience (and to limit the scope).

VI is a particular type of approximate inference technique that translates the problem of Bayesian inference into an optimization objective to be optimized w.r.t. the parameters of the approximate posterior. Interestingly, this objective is a lower bound to the logarithm of the aforementioned marginal likelihood, which enables hence convenient joint optimization over hyperparameters (that are not treated as random variables) in addition to the approximate posterior's parameters. VI does not only provide a principled way to identify approximate posterior processes via sparse GPs when the likelihood is Gaussian, but it also provides a solution to scenarios with arbitrary likelihoods and where the true posterior process is typically not a GP (such as in logistic regression). It turns out that the framework can be readily extended to multimodal likelihood problems, as well as multiregression and -classification problems where unknown noisy vector-valued functions need to be identified.

The remainder of this manuscript is organized as follows. In Section 2, we provide an overview over sparse GPs and recent extensions that enable further computational gains. Importantly, Section 2 only explains sparse GPs but outside the scope of approximate inference. This is subject of Section 3 giving a general background on VI where we refer to weight space models (such as deep neural networks) to ease the exposition. In Section 4, we combine the previous two sections and elucidate how to do VI with sparse GPs (that are function space models), but we also provide some tricks for practitioners. In the end, we cover some further reaching topics around sparse GPs and VI (Section 5) before concluding with a summary in Section 6.

2 Sparse Gaussian Processes

Informally, GPs can be imagined as a generalization of multivariate Gaussians that are indexed by a (possibly continuous) input domain rather than an index set. Exact and approximate inference techniques with GPs leverage conditioning operations that are conceptually equivalent to those in multivariate Gaussians. In Section 2.1, we therefore provide an overview of the most important conditioning operations in multivariate Gaussians and present their GP counterparts in Section 2.2. It turns out that these conditioning operations provide a natural way to express sparse GPs and generalize readily to interdomain GPs (Section 2.3), GPs with multiple outputs (Section 2.4) and deep GPs consisting of multioutput GPs stacked on top of one another (Section 2.5).

2.1 Multivariate Gaussian Identities for Conditioning

The identities presented in this section might evoke the impression of being a bit out of context at first sight but will turn out to be essential for understanding sparse GPs as presented in Section 2.2 and subsequent sections. We start by noting that conditionals of multivariate Gaussians are Gaussian as well. For that purpose, imagine a multivariate Gaussian $\mathcal{N}$ whose random variables are partitioned into two vectors $\mathbf{f}$ and $\mathbf{u}$ respectively. The joint distribution then assumes the following form:

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{u} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_{\mathbf{f}} \\ \boldsymbol{\mu}_{\mathbf{u}} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{\mathbf{ff}} & \boldsymbol{\Sigma}_{\mathbf{fu}} \\ \boldsymbol{\Sigma}_{\mathbf{uf}} & \boldsymbol{\Sigma}_{\mathbf{uu}} \end{pmatrix} \right), \tag{1}$$

where $\boldsymbol{\mu}_{\mathbf{f}}$ and $\boldsymbol{\mu}_{\mathbf{u}}$ refer to the marginal means of $\mathbf{f}$ and $\mathbf{u}$ respectively, and $\boldsymbol{\Sigma}_{\mathbf{ff}}$, $\boldsymbol{\Sigma}_{\mathbf{fu}}$, $\boldsymbol{\Sigma}_{\mathbf{uf}}$ and $\boldsymbol{\Sigma}_{\mathbf{uu}}$ to (cross-)covariance matrices. The conditional distribution of $\mathbf{f}$ given $\mathbf{u}$ can then be expressed as:

$$\mathbf{f}|\mathbf{u} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{f}} + \boldsymbol{\Sigma}_{\mathbf{fu}}\boldsymbol{\Sigma}_{\mathbf{uu}}^{-1}(\mathbf{u} - \boldsymbol{\mu}_{\mathbf{u}}), \boldsymbol{\Sigma}_{\mathbf{ff}} - \boldsymbol{\Sigma}_{\mathbf{fu}}\boldsymbol{\Sigma}_{\mathbf{uu}}^{-1}\boldsymbol{\Sigma}_{\mathbf{uf}} \right). \tag{2}$$

Let's imagine that other than the marginal distribution $p(\mathbf{u})$ with mean $\boldsymbol{\mu}_{\mathbf{u}}$ and covariance $\boldsymbol{\Sigma}_{\mathbf{uu}}$ from Equation (1), there is another Gaussian distribution over $\mathbf{u}$ with mean $\mathbf{m}_{\mathbf{u}}$ and covariance $\mathbf{S}_{\mathbf{uu}}$:

$$\mathbf{u} \sim \mathcal{N}\left(\mathbf{m}_{\mathbf{u}}, \mathbf{S}_{\mathbf{uu}} \right). \tag{3}$$

Denoting the distribution from Equation (2) as $p(\mathbf{f}|\mathbf{u})$, and the distribution from Equation (3) as $q(\mathbf{u})$, one obtains a marginal distribution over $\mathbf{f}$ as $q(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{u})q(\mathbf{u})\mathrm{d}\mathbf{u}$ that is again Gaussian:

$$\mathbf{f} \sim \mathcal{N}\left(\boldsymbol{\mu}_{\mathbf{f}} + \boldsymbol{\Sigma}_{\mathbf{fu}}\boldsymbol{\Sigma}_{\mathbf{uu}}^{-1}(\mathbf{m}_{\mathbf{u}} - \boldsymbol{\mu}_{\mathbf{u}}), \boldsymbol{\Sigma}_{\mathbf{ff}} - \boldsymbol{\Sigma}_{\mathbf{fu}}\boldsymbol{\Sigma}_{\mathbf{uu}}^{-1}(\boldsymbol{\Sigma}_{\mathbf{uu}} - \mathbf{S}_{\mathbf{uu}})\boldsymbol{\Sigma}_{\mathbf{uu}}^{-1}\boldsymbol{\Sigma}_{\mathbf{uf}} \right). \tag{4}$$

A quick sanity check reveals that if we had integrated $p(\mathbf{f}|\mathbf{u})$ with the marginal distribution $p(\mathbf{u})$ from Equation (1) instead of $q(\mathbf{u})$ from Equation (3), we would have recovered the original marginal distribution $p(\mathbf{f})$ with mean $\boldsymbol{\mu}_{\mathbf{f}}$ and covariance $\boldsymbol{\Sigma}_{\mathbf{ff}}$ from Equation (1) as expected.

Importantly, Equations (1) to (4) remain valid if we define $\mathbf{u}$ as $\mathbf{u} = \boldsymbol{\Phi}\mathbf{f}$ via a linear transformation $\boldsymbol{\Phi}$ of the random variable $\mathbf{f}$. In this case, since $\boldsymbol{\mu}_{\mathbf{f}}$ and $\boldsymbol{\Sigma}_{\mathbf{ff}}$ are given, the only remaining quantities to be identified are the mean $\boldsymbol{\mu}_{\mathbf{u}}$ and the (cross-)covariance matrices $\boldsymbol{\Sigma}_{\mathbf{fu}}$, $\boldsymbol{\Sigma}_{\mathbf{uf}}$ and $\boldsymbol{\Sigma}_{\mathbf{uu}}$, yielding:

$$\boldsymbol{\mu}_{\mathbf{u}} = \boldsymbol{\Phi}\boldsymbol{\mu}_{\mathbf{f}}, \tag{5}$$
$$\boldsymbol{\Sigma}_{\mathbf{fu}} = \boldsymbol{\Sigma}_{\mathbf{ff}}\boldsymbol{\Phi}^{\top} = (\boldsymbol{\Phi}\boldsymbol{\Sigma}_{\mathbf{ff}})^{\top} = \boldsymbol{\Sigma}_{\mathbf{uf}}^{\top}, \tag{6}$$
$$\boldsymbol{\Sigma}_{\mathbf{uu}} = \boldsymbol{\Phi}\boldsymbol{\Sigma}_{\mathbf{ff}}\boldsymbol{\Phi}^{\top}. \tag{7}$$

Note that the joint covariance matrix of $\mathbf{f}$ and $\mathbf{u}$ is degenerate (i.e. singular) because $\mathbf{u}$ is the result of a linear transformation of $\mathbf{f}$ and hence completely determined by $\mathbf{f}$.

2.2 Gaussian Processes and Conditioning

A GP represents a distribution denoted as $\mathcal{GP}$ over real-valued functions $f(\cdot) : \mathcal{X} \to \mathbb{R}$ defined over an input domain $\mathcal{X}$ that we assume to be continuous throughout this tutorial (although this does not need to be the case). While multivariate Gaussians represent a distribution over finite-dimensional vectors, GPs represent a distribution over uncountably infinite-dimensional functions—vector indexes in multivariate Gaussian random variables conceptually correspond to specific evaluation points $X \in \mathcal{X}$ in GP random functions. Formally, a GP is defined through two real-valued functions: a mean function $\mu(\cdot) : \mathcal{X} \to \mathbb{R}$ and a symmetric positive-definite covariance function $k(\cdot, \cdot') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a.k.a. kernel [Rasmussen and Williams, 2006]:

$$f(\cdot) \sim \mathcal{GP}\Big(\mu(\cdot), k(\cdot, \cdot')\Big), \tag{8}$$

where $\mu(\cdot) = \mathbb{E}[f(\cdot)]$ and $k(\cdot, \cdot') = \mathbb{E}[(f(\cdot) - \mu(\cdot))(f(\cdot') - \mu(\cdot'))] = \mathrm{cov}(f(\cdot), f(\cdot'))$ in accordance with multivariate Gaussian notation. Importantly, if we evaluate the GP at any finite subset $\{X_1, X_2, ..., X_N\}$ of $\mathcal{X}$ with cardinality N, we would obtain an N-dimensional multivariate Gaussian random variable $\mathbf{f}$:

$$\mathbf{f} \sim \mathcal{N}\Big(\boldsymbol{\mu_f}, \mathbf{K_{ff}}\Big), \tag{9}$$

where $\boldsymbol{\mu_f}$ and $\mathbf{K_{ff}}$ are the mean and covariance matrix obtained by evaluating the mean and covariance function respectively at $\{X_1, X_2, ..., X_N\}$, i.e. $\boldsymbol{\mu_f}[n] = \mu(X_n)$ and $\mathbf{K_{ff}}[n, m] = k(X_n, X_m)$. Here, square brackets conveniently refer to `numpy` indexing notation for vectors and matrices: $\boldsymbol{\mu_f}[n]$ refers to the entry at index n of the vector $\boldsymbol{\mu_f}$, and $\mathbf{K_{ff}}[n, m]$ refers to the entry at row index n and column index m of the matrix $\mathbf{K_{ff}}$. While the notation with subscript $\mathbf{f}$ puts emphasis on the random variable and is standard in the literature, it unfortunately hides away the explicit "dependence" on the evaluation points X_n which might confuse readers new to the subject.

Similarly to Equation (1) from the previous section on multivariate Gaussians, we can partition the uncountably infinite set of random variables represented by a GP into two sets: one that contains a finite subset denoted as $\mathbf{u}$ evaluated at a finite set of evaluation points $\{Z_1, Z_2, ..., Z_M\} \in \mathcal{X}$, such that $\mathbf{u}[m] = f(Z_m)$, with mean $\boldsymbol{\mu_u}$ and covariance matrix $\mathbf{K_{uu}}$; and one that contains the remaining uncountably infinite set of random variables denoted as $f(\cdot)$ evaluated at all locations in $\mathcal{X}$ except for the finitely many points Z_m:

$$\begin{pmatrix} f(\cdot) \\ \mathbf{u} \end{pmatrix} \sim \mathcal{GP}\left(\begin{pmatrix} \mu(\cdot) \\ \boldsymbol{\mu_u} \end{pmatrix}, \begin{pmatrix} k(\cdot, \cdot') & \mathbf{k_{\cdot u}} \\ \mathbf{k_{u \cdot'}} & \mathbf{K_{uu}} \end{pmatrix} \right), \tag{10}$$

where $\mathbf{k_{\cdot u}}$ and $\mathbf{k_{u \cdot'}}$ denote vector-valued functions that express the cross-covariance between the finite-dimensional random variable $\mathbf{u}$ and the uncountably infinite-dimensional random variable $f(\cdot)$, i.e. $\mathbf{k_{\cdot u}}[m] = k(\cdot, Z_m)$ and $\mathbf{k_{u \cdot'}}[m] = k(Z_m, \cdot')$. Note that $\mathbf{k_{\cdot u}}$ is row-vector-valued whereas $\mathbf{k_{u \cdot'}}$ is column-vector-valued, and that $\mathbf{k_{\cdot u}} = \mathbf{k_{u \cdot}^\top}$ holds.

One might ask why we have chosen the same notation $f(\cdot)$ in Equation (10) as in Equation (8) although both random variables have different evaluation domains ($\mathcal{X}$ without inducing points Z_m versus all of $\mathcal{X}$). The answer is a more careful notation could have been adopted [Matthews et al., 2016] but, technically, Equation (10) is correct and poses a degenerate distribution over $f(\cdot)$ and $\mathbf{u}$, because $\mathbf{u}$ is completely determined by $f(\cdot)$ (since $\mathbf{u}$ is the GP evaluated at the inducing points Z_m).

The conditional GP of $f(\cdot)$ conditioned on $\mathbf{u}$ corresponding to the joint from Equation (10) is obtained in a similar fashion as the conditional multivariate Gaussian from Equation (2) is obtained from the joint multivariate Gaussian in Equation (1), resulting in:

$$f(\cdot)|\mathbf{u} \sim \mathcal{GP}\Big(\mu(\cdot) + \mathbf{k_{\cdot u}} \mathbf{K_{uu}^{-1}}(\mathbf{u} - \boldsymbol{\mu_u}), k(\cdot, \cdot') - \mathbf{k_{\cdot u}} \mathbf{K_{uu}^{-1}} \mathbf{k_{u \cdot'}}\Big). \tag{11}$$

Similarly to the multivariate Gaussian case, one can assume another marginal distribution $q(\mathbf{u})$ over $\mathbf{u}$ as in Equation (3) (other than $p(\mathbf{u})$ which is the marginal distribution over $\mathbf{u}$ with mean $\boldsymbol{\mu_u}$ and covariance $\boldsymbol{\Sigma_{uu}}$ in accordance with Equation (10)). Denoting the conditional GP from Equation (11) as $p(f(\cdot)|\mathbf{u})$ and integrating out $\mathbf{u}$ with $q(\mathbf{u})$ yields $q(f(\cdot)) = \int p(f(\cdot)|\mathbf{u})q(\mathbf{u})\mathrm{d}\mathbf{u}$, which is a GP:

$$f(\cdot) \sim \mathcal{GP}\Big(\mu(\cdot) + \mathbf{k_{\cdot u}} \mathbf{K_{uu}^{-1}}(\mathbf{m_u} - \boldsymbol{\mu_u}), k(\cdot, \cdot') - \mathbf{k_{\cdot u}} \mathbf{K_{uu}^{-1}}(\mathbf{K_{uu}} - \mathbf{S_{uu}})\mathbf{K_{uu}^{-1}} \mathbf{k_{u \cdot'}}\Big), \tag{12}$$

and conceptually equivalent to its multivariate Gaussian counterpart from Equation (4). With Equation (12), we have arrived at the definition of a sparse GP as used in contemporary literature [Titsias,

2009]. In this context, the evaluation points Z_m are usually called "inducing points" or "inducing inputs" that refer to pseudo-training examples, and $\mathbf{u}$ is called "inducing variable" conceptually referring to noise-free pseudo-outputs observed at the inducing inputs.

The number M of inducing points Z_m governs the expressiveness of the sparse GP—more inducing points mean more pseudo-training examples and hence a more accurate approximate representation of a function posterior. However, since M determines the dimension of $\mathbf{u}$, more inducing points also mean higher memory requirements (as $\mathbf{K_{uu}}$ needs to be stored) and higher computational complexity (as $\mathbf{K_{uu}}$ needs to be inverted which is a cubic operation $\mathcal{O}(M^3)$). Practical limitations therefore incentivise low M explaining why the formulation is referred to as a "sparse" GP in the first place. Note that this line of reasoning assumes a naive approach for dealing with covariance matrices— there are more recent advances that enable approximate computations based on conjugate-gradient-type algorithms [Gardner et al., 2018], but this is outside the scope of this tutorial.

At this point, we need to highlight that the notation $q(f(\cdot))$ related to the approximate posterior process is mathematically sloppy since it colloquially refers to a distribution over functions for which no probability density exists. We will nevertheless continue with this notation occasionally— or with the notation $p(f(\cdot))$ to refer to the prior distribution over $f(\cdot)$—where we feel it makes the subject more easily digestible.

In practice, $q(f(\cdot))$ is used to approximate an intractable posterior process through variational inference—we will learn more about what this means in Section 4. In short and on a high level, variational inference phrases an approximate inference problem as an optimization problem where inducing points Z_m, as well as the mean $\mathbf{m_u}$ and the covariance $\mathbf{S_{uu}}$ of the inducing variable distribution $q(\mathbf{u})$, are treated as optimization arguments that are automatically identified in the course of training. The importance of Equation (12) is substantiated by the fact that it remains valid for interdomain GPs (Section 2.3) and multioutput GPs (Section 2.4), and forms the central building block of modern deep GPs (Section 2.5), all of which can be practically trained with variational inference.

2.3 Interdomain Gaussian Processes

In the preceding section, when introducing sparse GPs, we have defined $\mathbf{u}$ as a random variable obtained when evaluating the GP on a set of M inducing points $\{Z_1, Z_2, ..., Z_M\} \in \mathcal{X}$. In line with the second part of Section 2.1, we could have alternatively defined $\mathbf{u}$ differently as $\mathbf{u}[m] = \int f(X)\phi_m(X)\mathrm{d}X$ through a linear functional on $f(\cdot)$ with help of a set of "inducing features" $\{\phi_1(\cdot), \phi_2(\cdot), ..., \phi_M(\cdot)\}$ that are real-valued functions $\phi_m(\cdot) : \mathcal{X} \to \mathbb{R}$ [Lazaro-Gredilla and Figueiras-Vidal, 2009]. It turns out that in this case, Equation (12) remains valid and the only quantities to be identified are the mean for inducing variables $\boldsymbol{\mu_u}$ as well as (cross-)covariances $\mathbf{k_{\cdot u}}$ and $\mathbf{K_{uu}}$, which are computed as follows:

$$
\begin{aligned}
\boldsymbol{\mu_u}[m] &= \mathbb{E}\left[\int f(X)\phi_m(X)\,\mathrm{d}X\right] = \int \mathbb{E}\big[f(X)\big]\,\phi_m(X)\,\mathrm{d}X \\
&= \int \mu(X)\phi_m(X)\,\mathrm{d}X, \tag{13}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{k_{\cdot u}}[n] &= \mathbb{E}\left[\left(f(\cdot) - \mu(\cdot)\right)\left(\int (f(X') - \mu(X'))\phi_n(X')\,\mathrm{d}X'\right)\right] \\
&= \int \mathbb{E}\big[(f(\cdot) - \mu(\cdot))(f(X') - \mu(X'))\big]\,\phi_n(X')\,\mathrm{d}X' \\
&= \int k(\cdot, X')\phi_n(X')\,\mathrm{d}X', \tag{14}
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{K_{uu}}[m, n] &= \mathbb{E}\left[\left(\int (f(X) - \mu(X))\phi_m(X)\,\mathrm{d}X\right)\left(\int (f(X') - \mu(X'))\phi_n(X')\,\mathrm{d}X'\right)\right] \\
&= \int\int \mathbb{E}\big[(f(X) - \mu(X))(f(X') - \mu(X'))\big]\,\phi_m(X)\phi_n(X')\,\mathrm{d}X'\,\mathrm{d}X \\
&= \int\int k(X, X')\phi_m(X)\phi_n(X')\,\mathrm{d}X'\,\mathrm{d}X. \tag{15}
\end{aligned}
$$

We ask the reader at this stage to pause and carefully compare the definition of $\mathbf{u}$ in this section and Equations (13) to (15), with the definition of $\mathbf{u}$ at the end of Section 2.1 on multivariate Gaussians and Equations (5) to (7). Equations (5) to (7) can be rewritten in numpy indexing notation as:

$$\boldsymbol{\mu}_{\mathbf{u}}[m] \quad = \quad \sum_i \boldsymbol{\mu}_{\mathbf{f}}[i]\boldsymbol{\phi}_m[i], \tag{16}$$

$$\boldsymbol{\Sigma}_{\mathbf{fu}}[:,n] \quad = \quad \sum_j \boldsymbol{\Sigma}_{\mathbf{ff}}[:,j]\boldsymbol{\phi}_n[j], \tag{17}$$

$$\boldsymbol{\Sigma}_{\mathbf{uu}}[m,n] \quad = \quad \sum_i \sum_j \boldsymbol{\Sigma}_{\mathbf{ff}}[i,j]\boldsymbol{\phi}_m[i]\boldsymbol{\phi}_n[j]. \tag{18}$$

This provides a good intuition of how interdomain GPs relate to linear transformations of random variables in the multivariate Gaussian case: mean functions and covariance functions correspond to mean vectors and covariance matrices, inducing features to feature vectors, and integrals over a continuous input domain to sums over discrete indices. Mathematically, $\int f(X)\phi_m(X)\mathrm{d}X$ is a real-valued linear integral operator (which maps functions $f(\cdot)$ to real numbers) that generalizes the concept of a real-valued linear transformation operating on finite-dimensional vector spaces.

After obtaining a mathematical intuition for interdomain GPs and how they relate to linear transformations in multivariate Gaussians, it can be insightful to get a conceptual intuition with concrete examples. An important characteristic of the interdomain formulation is that inducing points can live in a domain different from the one in which the GP operates (which is $\mathcal{X}$), hence the naming. Of practical importance is also the question how to choose the features $\phi_m(\cdot)$ such that the covariance formulations from Equations (14) and (15) have closed-form expressions (the mean formulation from Equation (13) evaluates trivially to zero for a zero-mean function that assigns zero to every input location X, which is a common practical choice).

In the following, we present four examples: Dirac, Fourier, kernel-eigen and derivative features. Dirac features recover ordinary inducing points as a special case of the interdomain formulation. Fourier features enable inducing points to live in a frequency domain (different from $\mathcal{X}$ that is considered as time/space domain in this context). Kernel-eigen features are conceptually equivalent to principal components of a finite-dimensional covariance matrix but for uncountably-infinite dimensional kernel functions. Derivative features enable inducing points to evaluate the derivative of $f(\cdot)$ w.r.t. specific dimensions of X, as opposed to the the ordinary inducing point formulation that enables inducing points to evaluate $f(\cdot)$.

Dirac Features. Dirac features are trivially defined as $\phi_m(\cdot) = \delta_{Z_m}(\cdot)$ through the Dirac delta function $\delta_{Z_m}(\cdot)$ that puts all probability mass on Z_m. This makes inducing points live in $\mathcal{X}$ as expected, and $\boldsymbol{\mu}_{\mathbf{u}}$, $\mathbf{k}._{\mathbf{u}}$ and $\mathbf{K}_{\mathbf{uu}}$ recover the expressions from Section 2.2 for ordinary sparse GPs. Since Dirac features recover the ordinary inducing point formulation through a linear integral operator, it becomes clear that we can choose the same notation $f(\cdot)$ for both random variables in Equations (10) and (8), without worrying too much about one input domain being $\mathcal{X}$ excluding finitely many inducing points and the other one all of $\mathcal{X}$.

Fourier Features. Fourier features are defined as $\phi_m(\cdot) = \exp(-\mathrm{i}\boldsymbol{\omega}_m^\top \cdot)$ where $\boldsymbol{\omega}_m$ refers to an inducing frequency vector and i to the complex unit. Note that, practically, boundary conditions need to be defined for Fourier features. Otherwise, $\mathbf{K}_{\mathbf{uu}}$ would have infinite-valued entries on its diagonal for any stationary kernel (stationary kernels are a specific type of kernel where the covariance between two input locations X and X' only depends on the distance between X and X'). Also note that for specific stationary kernels, $\mathbf{k}._{\mathbf{u}}$ and $\mathbf{K}_{\mathbf{uu}}$ have real-valued closed-form expressions— see Hensman et al. [2018] for details. Fourier features underpin the term "interdomain" because the integral operator enables inducing points $\boldsymbol{\omega}_m$ to live in a frequency domain different from the time/space domain $\mathcal{X}$. While Hensman et al. [2018] chose fixed inducing frequencies $\boldsymbol{\omega}_m$ arranged in a grid-wise fashion, an interesting future research direction is to treat $\boldsymbol{\omega}_m$ as optimization arguments in the context of approximate inference (e.g. variational inference as explained later in Section 3). We need to stress that the original Fourier feature formulation from Hensman et al. [2018] does not use the L^2 inner product between $f(\cdot)$ and $\phi_m(\cdot)$ to define inducing variables $\mathbf{u}$ as we did at the beginning of this section, but introducing alternative ways how to define the inner product between two functions is outside the scope of this tutorial.

Kernel-Eigen Features. Kernel-eigen features can be given as $\phi_m(\cdot) = v_m(\cdot)$ where $v_m(\cdot)$ refers to the m-th eigenfunction of the kernel $k(\cdot, \cdot')$ with eigenvalue λ_m. Eigenfunctions of kernels are defined as $\int k(\cdot, X')v_m(X')\mathrm{d}X' = \lambda_m v_m(\cdot)$, similarly to eigenvectors of square matrices when replacing integrals over X' with sums over indexes. Equation (14) then trivially evaluates to:

$$\mathbf{k}_{\cdot\mathbf{u}}[n] = \int k(\cdot, X')v_n(X')\,\mathrm{d}X' = \lambda_n v_n(\cdot), \tag{19}$$

and is conceptually equivalent to rotating finite-dimensional vectors via principal component analysis. The covariance matrix from Equation (15) is computed as follows:

$$
\begin{aligned}
\mathbf{K}_{\mathbf{uu}}[m, n] &= \int \int k(X, X')v_m(X)v_n(X')\,\mathrm{d}X'\,\mathrm{d}X \\
&= \int v_m(X) \int k(X, X')v_n(X')\,\mathrm{d}X'\,\mathrm{d}X \\
&= \int v_m(X)\lambda_n v_n(X)\,\mathrm{d}X = \lambda_n \int v_m(X)v_n(X)\,\mathrm{d}X \\
&= \lambda_n \text{ if } m == n \text{ else } 0,
\end{aligned}
\tag{20}
$$

which is a diagonal matrix [Burt et al., 2019]. The step from the second to the third line merely utilizes the eigenfunction definition. The step from the third to the fourth line leverages that eigenfunctions are orthonormal, which means $\int v_m(X)v_n(X)\mathrm{d}X$ equals one if m equals n and is zero otherwise. This is of great practical importance since a diagonal $\mathbf{K}_{\mathbf{uu}}$ is more memory-efficient and invertible in linear rather than cubic time. Identifying eigenfunctions and eigenvalues for arbitrary kernels in closed-form is non-trivial, but solutions do exist for some [Rasmussen and Williams, 2006, Borovitskiy et al., 2020, Burt et al., 2020, Dutordoir et al., 2020a, Riutort-Mayol et al., 2020].

Derivative Features. In addition to the linear integral operator formulation from above in Equations (13) to (15) that contains Dirac, Fourier and kernel-eigen features as special cases, there is another possibility to define interdomain variables via the derivatives of a GP's function values [Adam et al., 2020, van der Wilk et al., 2020]. In this case, the inducing variable $\mathbf{u}$ is expressed as $\mathbf{u}[m] = \frac{\partial}{\partial X_{d(m)}} f(X)\big|_{X=Z_m}$ where Z_m refers to an inducing point and $d(m)$ to the input dimension of X over which the partial derivative is performed (as determined by the interdomain variable with index m). Every interdomain variable needs to specify a particular input dimension of X the derivative of $f(\cdot)$ is taken with respect to. Equations (13) to (15) then become:

$$\boldsymbol{\mu}_{\mathbf{u}}[m] = \frac{\partial}{\partial X_{d(m)}}\mu(X)\bigg|_{X=Z_m}, \tag{21}$$

$$\mathbf{k}_{\cdot\mathbf{u}}[n] = \frac{\partial}{\partial X'_{d(n)}}k(\cdot, X')\bigg|_{X'=Z_n}, \tag{22}$$

$$\mathbf{K}_{\mathbf{uu}}[m, n] = \frac{\partial^2}{\partial X_{d(m)}\partial X'_{d(n)}}k(X, X')\bigg|_{X=Z_m, X'=Z_n}, \tag{23}$$

presupposing differentiable mean and kernel functions. This can be useful, e.g. when $\mathcal{X}$ is a time domain and $f(\cdot)$ represents a space domain, but one seeks to express interdomain variables in a velocity domain [O'Hagan, 1992, Rasmussen and Williams, 2006]. Mathematically, the results above are not surprising since differentiation is a linear operator over function spaces.

2.4 Multioutput Gaussian Processes

A multioutput GP extends a GP to a distribution over vector-valued functions $\mathbf{f}(\cdot) : \mathcal{X} \to \mathbb{R}^D$, where D refers to the number of outputs [Alvarez et al., 2012]. While an ordinary GP outputs a scalar Gaussian random variable when evaluated at a specific input location $X \in \mathcal{X}$, a multioutput GP outputs a D-dimensional multivariate Gaussian random variable when evaluated at a specific input location X. Formally, a multioutput GP is defined similarly to an ordinary singleoutput GP as in Equation (8), but with a vector-valued mean function $\boldsymbol{\mu}(\cdot) : \mathcal{X} \to \mathbb{R}^D$ and a matrix-valued

cross-covariance function $\mathbf{K}(\cdot, \cdot') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{D \times D}$ [Miccnelli and Pontil, 2005]. Note that $\mathbf{K}(\cdot, \cdot')$ needs to output a proper cross-covariance matrix for every (X, X')-pair and must hence satisfy the following symmetry relation $\mathbf{K}(\cdot, \cdot') = \mathbf{K}(\cdot', \cdot)^\top$, i.e. $\mathbf{K}(\cdot, \cdot')[i, j] = \mathbf{K}(\cdot', \cdot)[j, i]$ in `numpy` indexing notation. We would like to remind the reader at this stage to not confuse multioutput notation with singleoutput notation from earlier for random vectors $\mathbf{f}$, mean vectors $\boldsymbol{\mu_f}$ and covariance matrices $\mathbf{K_{ff}}$ that result from evaluating a singleoutput GP in multiple locations $\{X_1, X_2, ..., X_N\}$.

One might ask how to evaluate a multioutput GP since this would naively lead to a random variable of extend $N \times D$ where N refers to the number of evaluation points. The answer is that we can flatten the $N \times D$ random variable into a vector of size ND. This results in a multivariate Gaussian random variable $\mathbf{f}$ as described in Equation (9) with a mean vector $\boldsymbol{\mu_f}$ of size ND and a covariance matrix $\mathbf{K_{ff}}$ of size $ND \times ND$. While the choice of flattening is up to the user, one could e.g. concatenate all D-dimensional random variables for each evaluation point. This yields a partitioned mean vector with N partitions of size D and a block-covariance matrix with $N \times N$ blocks each holding a $D \times D$ matrix. The block covariance matrix is required to form a proper covariance matrix, i.e. it needs to be symmetric (which is guaranteed by the imposed symmetry relation from the previous paragraph).

The "flattening trick" already hints at the fact that a multioutput GP can be indeed defined differently as a distribution over real-valued functions with a real-valued mean function and a real-valued covariance function. This is made possible through the "output-as-input" view where a multioutput GP's input domain is extended through an index set $\mathcal{I} = \{1, 2, ..., D\}$ to index output dimensions [van der Wilk et al., 2020]. More formally, this yields $f(\cdot) : (\mathcal{X}, \mathcal{I}) \to \mathbb{R}$ distributed according to the mean function $\mu(\cdot) : (\mathcal{X}, \mathcal{I}) \to \mathbb{R}$ and the covariance function $k(\cdot, \cdot') : (\mathcal{X}, \mathcal{I}) \times (\mathcal{X}, \mathcal{I}) \to \mathbb{R}$, where the dot notation $\cdot$ now refers to a pair (the first element of the pair being the input $X \in \mathcal{X}$ and the second the output index $i \in \mathcal{I}$). In this notation, the kernel only needs to satisfy the same symmetry relation as an ordinary singleoutput kernel and Equation (8) for singleoutput GPs remains applicable (under the extended input domain).

The advantage of defining a multioutput GP this way is that it makes the evaluation at arbitrary input-output-index pairs (X, i) more convenient, i.e. it is not required to evaluate the multioutput GP at all outputs i for a specific input X. The latter comes in handy e.g. for (approximate) Bayesian inference in multilabel problems (that have multidimensional targets) when training examples can have missing labels (i.e. the set of labels for some X can be incomplete).

The output-as-input view not only ensures that Equation (8) remains valid for multioutput GPs but also ensures that Equation (12) remains valid for sparse multioutput GPs under an output-index-extended input domain. Naively, one could specify MD inducing points Z (or inducing features $\phi(\cdot)$ in the interdomain formulation) and MD inducing variables $\mathbf{u}$—M for each output head respectively. In Equation (12), this would lead $\boldsymbol{\mu_u}$ and $\mathbf{m_u}$ to be MD-dimensional vectors, $\mathbf{K_{uu}}$ and $\mathbf{S_{uu}}$ to be $MD \times MD$ matrices, and $\mathbf{k_{\cdot u}}$ and $\mathbf{k_{u \cdot'}}$ to be MD-dimensional vector-valued functions.

At this point, we have acquired an understanding of sparse multioutput GPs, but it can be insightful to continue with a discussion on computational efficiency and how to design computationally efficient multioutput GPs in practice. The biggest computational chunk in Equation (12) is the inversion of $\mathbf{K_{uu}}$ which poses a cubic operation in MD—i.e. $\mathcal{O}(M^3 D^3)$—under the naive specification from earlier. This can be addressed two-fold: by a separate independent multioutput GP that sacrifices the ability to model output correlations for the sake of computational efficiency [van der Wilk et al., 2020], or by squashing a latent separate independent multioutput GP through a linear transformation to couple outputs—the latter can be achieved with a linear model of coregionalization [Journel and Huijbregts, 1978] or convolutional GPs [Alvarez et al., 2010, van der Wilk and Rasmussen, 2017].

Separate Independent Multioutput GP. A separate independent multioutput GP specifies D separate singleoutput GPs—one for each output dimension—and specifies different output dimensions to have zero-covariance, i.e. $\mathbf{K}(\cdot, \cdot')$ is a diagonal matrix-valued function under the output-as-output view from earlier. Under the MD-flatting outlined previously, a separate independent multioutput sparse GP makes $\mathbf{K_{uu}}$ be a block matrix with $M \times M$ blocks each of which contains a diagonal matrix of size $D \times D$. However, we could have alternatively chosen a DM-flattening view rather than an MD-flattening view, in which case $\mathbf{K_{uu}}$ would be a block-diagonal matrix with $D \times D$ blocks, where each diagonal block is a full $M \times M$ matrix but each off-diagonal block contains only zeros. The latter rearrangement enables to invert $\mathbf{K_{uu}}$ by inverting the D diagonal blocks (each of size $M \times M$) separately, yielding an improved computational complexity of $\mathcal{O}(M^3 D)$.

Linear Model of Coregionalization.   A linear model of coregionalization [Journel and Huijbregts, 1978] provides a simple approach to construct a multioutput GP that is both computationally efficient and ensures correlated output heads. Resorting back to the output-as-output view, the idea is to squash a latent separate independent multioutput GP with a diagonal matrix-valued kernel $\mathbf{K}_g(\cdot, \cdot')$ through a linear transformation [Dutordoir et al., 2018]. The linear transformation $\mathbf{W}$ is defined to be a $D \times D_g$ matrix where D_g denotes the number of latent outputs. Since D_g is user-defined, it enables control over computational complexity because it determines the computational burden of inverting latent covariance matrices. This allows for efficient fully-correlated multioutput GPs with high-dimensional outputs where $d \gg d_g$. Denoting the latent GP in the output-as-output view as:

$$\mathbf{g}(\cdot) \ \sim \ \mathcal{GP}\Big(\boldsymbol{\mu}_g(\cdot), \mathbf{K}_g(\cdot, \cdot')\Big), \tag{24}$$

where $\mathbf{g}(\cdot)$ refers to the latent vector-valued random function and $\boldsymbol{\mu}_g(\cdot)$ to the latent vector-valued mean function, a proper multioutput GP can be obtained via $\mathbf{f}(\cdot) = \mathbf{W}\mathbf{g}(\cdot)$, resulting in:

$$\mathbf{f}(\cdot) \ \sim \ \mathcal{GP}\Big(\mathbf{W}\boldsymbol{\mu}_g(\cdot), \mathbf{W}\mathbf{K}_g(\cdot, \cdot')\mathbf{W}^\top\Big). \tag{25}$$

Rather than defining the latent multioutput GP with Equation (24), we could have alternatively used the corresponding output-as-output view of a sparse multioutput GP according to Equation (12) in combination with the kernel function $\mathbf{K}_g(\cdot, \cdot')$, in which case Equation (25) would have represented a fully correlated but sparse multioutput GP with efficient matrix inversion in latent space. A more detailed discussion on the topic can be found once more in van der Wilk et al. [2020].

Convolutional GP.   In similar vein to a linear model of coregionalization, we can construct coupled output heads from a latent separate independent multioutput GP as given in Equation (24) via a convolutional GP. The idea is to define $\mathbf{f}(\cdot) = \int \mathbf{G}(\cdot - X)\mathbf{g}(X)\mathrm{d}X$ where $\mathbf{G}(\cdot)$ is a matrix-valued function that outputs a $D \times D_g$ matrix for every input X. The corresponding process over $\mathbf{f}(\cdot)$ is a proper multioutput GP because of the linearity of the convolution operator, and is given by:

$$\mathbf{f}(\cdot) \ \sim \ \mathcal{GP}\Big(\int \mathbf{G}(\cdot - X)\boldsymbol{\mu}_g(X)\,\mathrm{d}X, \int\int \mathbf{G}(\cdot - X)\mathbf{K}_g(X, X')\mathbf{G}^\top(\cdot' - X')\,\mathrm{d}X'\,\mathrm{d}X\Big), \tag{26}$$

where the matrix-valued function $\mathbf{G}(\cdot)$ is usually chosen such as to yield tractable integrals. The presentation of convolutional GPs here follows the descriptions in Alvarez et al. [2010] and van der Wilk et al. [2020], but similar models have been proposed in earlier work [Higdon, 2002, Boyle and Frean, 2004, Alvarez and Lawrence, 2008, Alvarez et al., 2009]. Note that Equation (26) builds upon a latent GP according to Equation (24) for conceptual convenience, but we could have used a sparse GP according to Equation (12) instead.

Image-Convolutional GP.   Despite the similarity in naming, an image-convolutional GP is different from a convolutional GP. Let's imagine a domain of images and that a single image X is subdivided into a set of (possibly overlapping) patches, all of equal size and indexed by p. For notational convenience, we define the p-th patch of X as $X[p]$. We can then define an ordinary singleoutput GP operating in a latent patch space with random functions denoted as $g(\cdot[p])$ and with mean function $\mu_g(\cdot[p])$ and kernel $k_g(\cdot[p], \cdot'[p'])$, where the notation $\cdot[p]$ refers to the p-th patch of an input image and where the input image is indicated by the usual dot notation $\cdot$. It turns out that this latent singleoutput GP defined in patch space induces a multioutput GP over vector-valued functions $\mathbf{f}(\cdot)$ that operates in image space (where the number of outputs equals the number of patches). The vector-valued random function $\mathbf{f}(\cdot)$ then relates to the latent real-valued function $g(\cdot[p])$ as $\mathbf{f}(\cdot)[p] = g(\cdot[p])$. Similarly, the multioutput mean function can be described as $\boldsymbol{\mu}(\cdot)[p] = \mu_g(\cdot[p])$ and the multioutput kernel as $\mathbf{K}(\cdot, \cdot')[p, p'] = k_g(\cdot[p], \cdot'[p'])$. This design was inspired by convolutional neural networks and a first description can be found in van der Wilk and Rasmussen [2017] which was later extended to deep architectures by Blomqvist et al. [2019] and Dutordoir et al. [2020b]. The difference between a convolutional and an image-convolutional GP is hence that the former performs a convolution operation on the input domain, whereas the latter performs an operation that resembles a discrete two-dimensional convolution on a single element from an image input domain.

Derivative GP.   We conclude this section with derivative GPs because they provide a natural example of multioutput GPs. Earlier, we have seen how to define interdomain variables that are partial derivatives of a singleoutput GP's random function (evaluated at specific locations). It turns out

that for any singleoutput GP with mean function $\mu(\cdot)$ and kernel $k(\cdot, \cdot')$, the derivative of $f(\cdot)$ w.r.t. X gives rise to a proper multioutput GP where the number of output dimensions D equals the dimension of $\mathcal{X}$—presupposing that $\mu(\cdot)$ and $k(\cdot, \cdot')$ are differentiable. More precisely, in the output-as-output view, we obtain $\mathbf{f}(\cdot) = \nabla_X f(X)\big|_{X=\cdot}$ with the corresponding multioutput mean function $\boldsymbol{\mu}(\cdot) = \nabla_X \mu(X)\big|_{X=\cdot}$ and the multioutput kernel function $\mathbf{K}(\cdot, \cdot') = \nabla_X \nabla_{X'} k(X, X')\big|_{X=\cdot, X'=\cdot'}$. Analysing the derivative of a multioutput GP, the resulting random function and its mean function are matrix-valued—in Jacobian notation $\mathbf{J_f}(\cdot)$ and $\mathbf{J_{\mu}}(\cdot)$—, and the kernel function is a hypercubical tensor of dimension four (we refrain from the mathematical notation for preserving a clear view). However, by applying the flattening trick from earlier, we can again obtain a multioutput GP in the output-as-output view—e.g. by flattening the matrix-valued random function $\mathbf{J_f}(\cdot)$ and its matrix-valued mean function $\mathbf{J_{\mu}}(\cdot)$ into vector-valued functions, which induces a flattening of the four-dimensional hypercubical tensor-valued kernel function into a matrix-valued kernel function.

2.5 Deep Gaussian Processes

A deep GP is obtained by stacking multioutput GPs on top of each other [Damianou and Lawrence, 2013]. The output of one GP determines where the next GP is evaluated, i.e. the output dimension (= number of outputs) of the GP below needs to conform with the input dimension of the GP on top. More formally, imagine L multioutput GPs with random vector-valued functions denoted in the output-as-output notation as $\{\mathbf{f}^{(1)}(\cdot), \mathbf{f}^{(2)}(\cdot), ..., \mathbf{f}^{(L)}(\cdot)\}$. A single input X is propagated through the deep GP as follows. The first GP with index 1 is evaluated at X yielding a vector-valued random variable $\mathbf{f}^{(1)}$. Drawing a single sample from $\mathbf{f}^{(1)}$ determines where to evaluate the second GP with index 2. This yields another vector-valued random variable $\mathbf{f}^{(2)}$ that determines where to evaluate the third GP, and so forth. This process is repeated until we arrive at the last GP with index L—the random variable $\mathbf{f}^{(L)}$ is considered as output of the deep GP for input X. The graphical model illustrating this process is depicted in Figure 1.
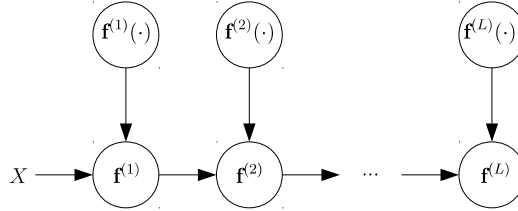

Figure 1: Graphical model of a deep GP. A single input X determines where to evaluate the first GP over random vector-valued functions $\mathbf{f}^{(1)}(\cdot)$ yielding a multivariate random variable $\mathbf{f}^{(1)}$. A single sample from $\mathbf{f}^{(1)}$ determines where to evaluate the next GP over $\mathbf{f}^{(2)}(\cdot)$ resulting in $\mathbf{f}^{(2)}$. Repeating this process until the last GP with index L yields $\mathbf{f}^{(L)}$, a sample of which is one final output for X.

Note that for deep GPs, the notation $\mathbf{f}$ refers to a vector-valued random variable as a result of evaluating all output heads of a multioutput GP at a single input location X. This clashes with the same notation used for shallow singleoutput GPs where $\mathbf{f}$ refers to a vector-valued random variable as a result of evaluating a singleoutput GP in multiple locations $\{X_1, X_2, ..., X_N\}$. We apologize for the confusion but there are only so many ways to represent vector-valued quantities. Also note that if we propagate N samples $\{X_1, X_2, ..., X_N\}$ through a deep GP (instead of just one as depicted above), we would need to evaluate the first GP at all N inputs yielding a random variable of size $N \times D^{(1)}$ in the output-as-output view (where $D^{(1)}$ refers to the number of outputs of the first GP). Sampling precisely once from this $N \times D^{(1)}$ random variable results in N vector-valued samples of size $D^{(1)}$ each, that would be used to evaluate the second GP, and so on. The final output would be a random variable of size $N \times D^{(L)}$, where $D^{(L)}$ refers to the number of outputs of the last GP.

The behavior of a deep GP is illustrated in Figure 2 for singleoutput GP building blocks that have one-dimensional input domains (for reasons of interpretability). With increasing depth, function values become more narrow-ranged and the function's behavior changes from smoothly to more abruptly [Duvenaud et al., 2014]. The latter is not surprising since once samples at intermediate layers are mapped to similar function values, they won't assume very different function values in

subsequent layers. While this enables to potentially model challenging functions that are less smooth (which may be difficult with an ordinary shallow GP), the marginal distributions over function values in every layer (except for the first one of course) are no longer Gaussian and hence impede analytical uncertainty estimates, which is usually considered a hallmark of GP models.
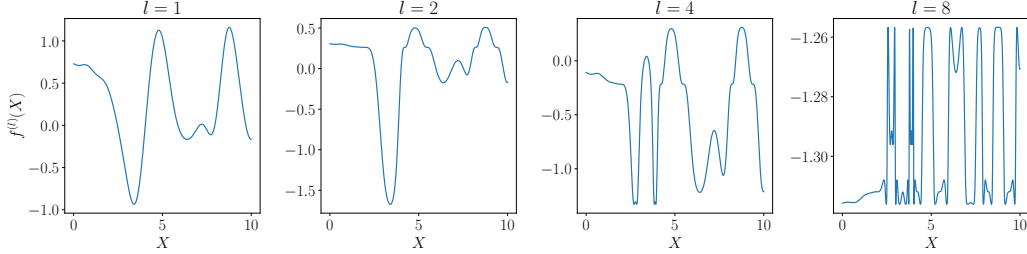

Figure 2: Samples from a deep GP at various depths. The deep GP building block for this figure is a singleoutput GP operating on a one-dimensional input domain and using an RBF kernel (lengthscale 0.7) with a zero-mean function. An equally spaced set of evaluation points X from the interval $[0, 10]$ is propagated through the deep GP and the function values $f^{(l)}(X)$ are shown at increasing depths ($l \in \{1, 2, 4, 8\}$). With depth increasing, function values rapidly change and assume values from a narrow range. The reason is that once samples are mapped to similar values in intermediate layers, they won't assume significantly different values in subsequent layers—see Duvenaud et al. [2014].

We conclude this section by noting that the definition of a deep GP is independent of the type of shallow GP used as a building block. Modern deep GPs stack sparse GPs, as presented in Equation (12), on top of each other to be computationally efficient [Salimbeni and Deisenroth, 2017, Salimbeni et al., 2019]. Up to this point, we haven't yet addressed how to do inference in (deep) sparse GPs. The reason is that exact inference is not possible and one has to resort to approximate inference techniques. Variational inference (VI) is a convenient tool used by contemporary literature in this context. We therefore dedicate the next section (Section 3) to explain general VI, before coming back to VI in shallow and deep sparse GPs later on in Section 4.

3 Variational Inference

VI is a specific type of approximate Bayesian inference. As the name indicates, approximate Bayesian inference deals with approximating posterior distributions that are computationally intractable. The goal of this section is to provide an overview of VI, where we resort mostly to the parameter space view and where we assume a supervised learning scenario. How to combine sparse GPs (that are function space models) with VI is subject of Section 4 later on. We begin with Section 3.1, where we derive vanilla VI. In Section 3.2, we demonstrate an alternative way to derive VI with importance weighting that provides a more accurate solution at the expense of increased computational complexity. After that, in Section 3.3, we introduce latent-variable VI to enable more flexible models. In Section 3.4, we combine latent-variable VI with importance weighting to trade computational cost for accuracy. Finally, in Section 3.5, we present how to do VI in a hierarchical and compositional fashion giving rise to a generic framework for Bayesian deep learning via the concept of Bayesian layers [Tran et al., 2019].

3.1 Vanilla Variational Inference

Let's start be revisiting Bayesian inference for supervised learning. Imagine some input X, some observed variable y and the probability of observing y given X via a parametric distribution $p_\gamma(y|\theta, X)$ with hyperparameters γ and unknown parameters θ. Our goal is to infer θ and we have some prior belief over θ through the distribution $p_\gamma(\theta)$, where we assume for notational convenience that both $p_\gamma(y|\theta, X)$ and $p_\gamma(\theta)$ are hyperparameterized by γ. Inference over θ is then obtained via the posterior distribution over θ after observing y and X:

$$p_\gamma(\theta|y, X) = \frac{p_\gamma(y|\theta, X)p_\gamma(\theta)}{\int p_\gamma(y|\theta, X)p_\gamma(\theta)\mathrm{d}\theta}, \tag{27}$$

A) Variational Inference

θ or $f(\cdot)$ θ or $f(\cdot)$

$X \rightarrow y$ $X_n \rightarrow y_n$

$n = \{1, 2, ..., N\}$

general formulation   i.i.d. dataset

B) Latent-Variable Variational Inference

θ or $f(\cdot)$ θ or $f(\cdot)$

h h_n

$X \rightarrow y$ $X_n \rightarrow y_n$

$n = \{1, 2, ..., N\}$

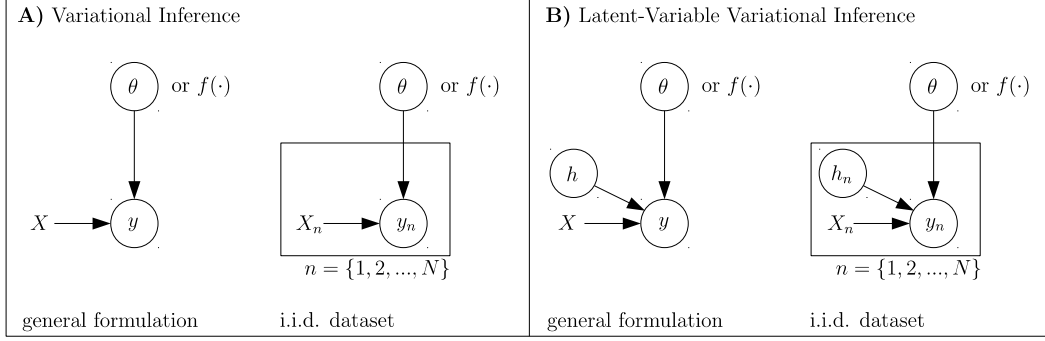general formulation   i.i.d. dataset

Figure 3: Graphical models for VI in supervised learning settings. X refers to inputs and y to labels. Unknown functions are depicted via the variable θ (in parameter space view) or $f(\cdot)$ (in function space view). Vanilla VI is illustrated in **A)** and subject of Section 3.1, whereas latent-variable VI is illustrated in **B)** and subject of Section 3.3. Latent-variable VI introduces another latent variable h (in addition to θ) to enable more expressive generative models. For the sake of clarity, we present in both cases one general formulation and another one in which the dataset is i.i.d. which is a common assumption in most scenarios—individual training examples are then indexed with n.

where $p_\gamma(y|\theta, X)$ is referred to as likelihood, $p_\gamma(\theta)$ as prior and $p_\gamma(y|X) = \int p_\gamma(y|\theta, X)p_\gamma(\theta)\mathrm{d}\theta$ as marginal likelihood (or evidence). The corresponding graphical model for this inference problem is depicted in Figure 3 A) on the left-hand side (denoted as "general formulation"). The graphical model represents the joint distribution $p_\gamma(y|\theta, X)p_\gamma(\theta)$ between y and θ given X, which is also referred to as "generative model". The challenge in computing the posterior is that the marginal likelihood usually does not have a closed form solution except for special cases, e.g. when the prior is conjugate to the likelihood (which we won't consider in this tutorial). When the marginal likelihood does have a closed form solution, it is usually maximized w.r.t. to the hyperparemeters γ of the generative model before the exact posterior is computed [Bishop, 2006, Rasmussen and Williams, 2006]. Note that the hyperparameters γ are also referred to as "generative parameters".

The idea in VI is to introduce an approximation $q_\psi(\theta)$, parameterized via ψ, to the intractable posterior $p_\gamma(\theta|y, X)$, and to optimize for ψ such that the approximate posterior becomes close to the true posterior. In this regard, the approximate posterior $q_\psi(\theta)$ is also referred to as "variational model" and ψ as "variational parameters". The question is which optimization objective to choose to identify optimal variational parameters ψ. We are going to respond to this question shortly but for now, we commence with the negative Kullback-Leibler divergence (KL) between the approximate and the true posterior, which can be written as (by applying Bayes' rule to the true posterior):

$$-\mathrm{KL}\Big(q_\psi(\theta)\Big|\Big|p_\gamma(\theta|y, X)\Big) = \int q_\psi(\theta) \ln p_\gamma(y|\theta, X) \, \mathrm{d}\theta - \mathrm{KL}\Big(q_\psi(\theta)\Big|\Big|p_\gamma(\theta)\Big) - \ln p_\gamma(y|X). \quad (28)$$

Rearranging by bringing the log marginal likelihood term $\ln p_\gamma(y|X)$ to the left yields:

$$\ln p_\gamma(y|X) - \mathrm{KL}\Big(q_\psi(\theta)\Big|\Big|p_\gamma(\theta|y, X)\Big) = \underbrace{\int q_\psi(\theta) \ln p_\gamma(y|\theta, X) \, \mathrm{d}\theta - \mathrm{KL}\Big(q_\psi(\theta)\Big|\Big|p_\gamma(\theta)\Big)}_{=:\mathrm{ELBO}(\gamma,\psi)}. \quad (29)$$

The term on the right-hand side is referred to as the evidence lower bound $\mathrm{ELBO}(\gamma, \psi)$ [Rasmussen and Williams, 2006] since it poses a lower bound to the log marginal likelihood (a.k.a. log evidence)—"log evidence lower bound" might hence be a more suitable description but omitting "log" is established convention. The ELBO is a lower bound because the KL between the posterior approximation and the true posterior is non-negative. Since the log marginal likelihood does not depend on the variational parameters ψ, the ELBO assumes its maximum when the approximate posterior equals the true one, i.e. $q_\psi(\theta) = p_\gamma(\theta|y, X)$, in which case the KL term on the left is zero and the ELBO recovers the log marginal likelihood exactly.

Note how the formulation for the ELBO does not require to know the true posterior in its functional form a priori in order to identify an optimal approximation, because Equation (29) was obtained via decomposing the intractable posterior via Bayes' rule. Also note that the log marginal likelihood is

usually the preferred objective to maximize for the generative hyperparameters γ, as mentioned earlier. Contemporary VI methods therefore maximize the evidence lower bound $\max_{\gamma, \psi} \text{ELBO}(\gamma, \psi)$ jointly w.r.t. both generative parameters γ and variational parameters ψ. Some current methods with deep function approximators choose a slight modification of Equation (29) by multiplying the KL term between the approximate posterior and the prior with a positive β-parameter. This is called "β-VI" and recovers a maximum expected log likelihood objective as a special case when $\beta \to 0$. It has been proposed by Higgins et al. [2017], and Wenzel et al. [2020] provide a recent discussion.

Assuming an optimal approximate posterior has been identified after optimizing the ELBO w.r.t. both variational parameters ψ and generative parameters γ, the next question is how to use it, namely how to predict $y^\star$ for a new data point $X^\star$ that is not part of the training data. The answer is:

$$p(y^\star|X^\star) = \int p_\gamma(y^\star|\theta, X^\star) q_\psi(\theta) \, \mathrm{d}\theta, \tag{30}$$

by forming the joint between the likelihood $p_\gamma(y^\star|\theta, X^\star)$ and the approximate posterior $q_\psi(\theta)$, and integrating out θ. If the integration has no closed form, one has to resort to Monte Carlo methods—i.e. replace the integral over θ with an empirical average via samples obtained from $q_\psi(\theta)$.

So far, we haven't made any assumptions on how the generative model looks like precisely. In supervised learning, it is however common to assume an i.i.d. dataset in the sense that the training set comprises N i.i.d. training examples in the form of (X_n, y_n)-pairs. The corresponding graphical model is depicted in Figure 3 A) on the right denoted as "i.i.d. dataset". In this case, the likelihood is given by $\prod_{n=1}^{N} p_\gamma(y_n|\theta, X_n)$ and the ELBO becomes:

$$\text{ELBO}(\gamma, \psi) = \sum_{n=1}^{N} \int q_\psi(\theta) \ln p_\gamma(y_n|\theta, X_n) \, \mathrm{d}\theta - \text{KL}\Big(q_\psi(\theta)\Big|\Big|p_\gamma(\theta)\Big). \tag{31}$$

An interesting fact to note is that in case of large N, Equation (31) can be approximated by Monte Carlo using minibatches, which can be used for parameter updates without exceeding potential memory limits [Hensman et al., 2013]. The corresponding predictions $\{y_n^\star\}_{n=1,..,N^\star}$ for new data points $\{X_n^\star\}_{n=1,..,N^\star}$ in the i.i.d. setting are:

$$p(y_1^\star, ..., y_{N^\star}^\star|X_1^\star, ..., X_{N^\star}^\star) = \int \prod_{n=1}^{N^\star} p_\gamma(y_n^\star|\theta, X_n^\star) q_\psi(\theta) \, \mathrm{d}\theta. \tag{32}$$

Note that we have deliberately not made any assumptions on the dimensions of y, X, θ, γ and ψ to keep the notation light (which doesn't mean that these quantities need to be scalars). We also chose the weight space view by using θ instead of the function space view, although both are conceptually equivalent. The function space view can be obtained by replacing θ with $f(\cdot)$ in all formulations and equations above. Practically, one would need to be careful with expectations and KL divergences between infinite-dimensional random variables. We are going to address this issue later in Section 4 when talking about VI in sparse GPs (that naturally assume the function space view).

Since the probability distributions have been held generic so far, it can be insightful to provide some examples. To this end, imagine an i.i.d. regression problem with one-dimensional labels. Assume the prior $p(\boldsymbol{\theta})$ is a mean field multivariate Gaussian over the vectorized weights $\boldsymbol{\theta}$ in a neural network, with mean vector $\boldsymbol{\mu_\theta}$ and variance vector $\boldsymbol{v_\theta}$. Let the likelihood $p_\gamma(y_n|\boldsymbol{\theta}, X_n)$ be a homoscedastic Gaussian with variance $v_{\text{lik}}^{(\gamma)}$, whose mean depends on the neural net's output—i.e. $\mu_{\text{lik}}(X_n) = f_{\boldsymbol{\theta}}(X_n)$ where $f_{\boldsymbol{\theta}}(X_n)$ denotes the output of the neural net for the input X_n. In this context, the superscript (γ) marks the likelihood variance as a generative parameter. The variational approximation $q_\psi(\boldsymbol{\theta})$ could then be a mean field multivariate Gaussian as well, with mean vector $\mathbf{m}_{\boldsymbol{\theta}}^{(\psi)}$ and variance vector $\mathbf{v}_{\boldsymbol{\theta}}^{(\psi)}$, where the superscript (ψ) marks variational parameters. We have just arrived at a vanilla Bayesian neural network as in Blundell et al. [2015].

We can increase the expressiveness of the likelihood by making it heteroscedastic, i.e. by letting the neural net output a two-dimensional vector $\mathbf{f}_{\boldsymbol{\theta}}(\cdot)$ instead of a scalar to encode both the mean and the variance. This is achieved by defining $\mu_{\text{lik}}(X_n) = \mathbf{f}_{\boldsymbol{\theta}}(X_n)[1]$ and $v_{\text{lik}}(X_n) = g(\mathbf{f}_{\boldsymbol{\theta}}(X_n)[2])$ where 1 and 2 index both network outputs and $g(\cdot)$ is a strictly positive function (because the neural net's output is unbounded in general). In the latter case, there wouldn't be any generative parameter γ anymore because the likelihood variance has become a function of the neural net's output.

In practice, during optimization with gradient methods, the reparameterization trick [Kingma and Welling, 2014, Rezende et al., 2014] is applied to the random variable $\boldsymbol{\theta}$ in order to establish a differentiable relationship between $\boldsymbol{\theta}$ and the parameters of the distribution $q_\psi(\boldsymbol{\theta})$ from which $\boldsymbol{\theta}$ is sampled, i.e. the mean and the variance parameters $\mathbf{m}_{\boldsymbol{\theta}}^{(\psi)}$ and $\mathbf{v}_{\boldsymbol{\theta}}^{(\psi)}$. This is known to produce parameter updates with lower variance leading to better optimization. If we replace the prior and the approximate posterior with (multioutput) GPs and replace the notation $f_{\boldsymbol{\theta}}(\cdot)$ and $\mathbf{f}_{\boldsymbol{\theta}}(\cdot)$ with $f(\cdot)$ and $\mathbf{f}(\cdot)$ accordingly in the likelihood, we would obtain the GP equivalents of the homoscedastic and heteroscedastic Bayesian neural networks respectively. However, in GPs, one usually treats certain kernel hyperparameters as generative parameters γ, which means the prior is subject to optimization as opposed to the Bayesian neural network case.

We conclude by mentioning a related approximate inference scheme called expectation propagation (EP) [Bishop, 2006, Bui et al., 2017] that also encourages an approximate posterior to be close to the true posterior via a KL objective, similar to VI. In fact, EP chooses a similar objective as in Equation (28) but with swapped arguments in the KL. The practical difference is that VI tends to provide mode-centered solutions whereas EP tends to provide support-covering solutions at the price of potentially significant mode mismatch [Bishop, 2006] (if the approximation is unimodal but the true posterior multimodal). There are at least two more advantages of VI over vanilla EP. First, in VI, the expectation is w.r.t. to the approximate posterior and hence amenable to sampling and stochastic optimization with gradient methods, whereas the expectation in EP is w.r.t. to the unknown optimal posterior. And second, VI is principled in that it lower-bounds the log marginal likelihood therefore encouraging convenient optimization not only over variational but also generative parameters.

3.2 Importance-Weighted Variational Inference

Importance-weighted VI provides a way of lower-bounding the log marginal likelihood more tightly and with less estimation variance at the expense of increased computational complexity. We start by showing that there is an alternative way to derive the ELBO, in addition to the derivation from the previous section, according to the following formulation:

$$
\begin{aligned}
\ln p_\gamma(y|X) &= \ln \int p_\gamma(y|\theta, X) p_\gamma(\theta)\, \mathrm{d}\theta = \ln \int q_\psi(\theta) \frac{p_\gamma(y|\theta, X) p_\gamma(\theta)}{q_\psi(\theta)}\, \mathrm{d}\theta \quad (33)\\
&\geq \int q_\psi(\theta) \ln \frac{p_\gamma(y|\theta, X) p_\gamma(\theta)}{q_\psi(\theta)}\, \mathrm{d}\theta = \mathrm{ELBO}(\gamma, \psi), \quad (34)
\end{aligned}
$$

where the inequality comes from applying Jensen's inequality that swaps the logarithm with the expectation over $q_\psi(\theta)$. While this derivation is straightforward, it has the disadvantage of only demonstrating that the ELBO lower-bounds the log marginal likelihood but not by how much, namely the KL between the approximate and the true posterior, as shown in Equation (29).

In order to obtain an importance-weighted formulation that bounds the log marginal likelihood more tightly [Burda et al., 2016, Domke and Sheldon, 2018], we need to proceed from Equation (33) before applying Jensen:

$$
\begin{aligned}
\ln p_\gamma(y|X) &= \ln \int q_\psi(\theta) \frac{p_\gamma(y|\theta, X) p_\gamma(\theta)}{q_\psi(\theta)}\, \mathrm{d}\theta = \ln \mathbb{E}_{q_\psi(\theta)}\left[\frac{p_\gamma(y|\theta, X) p_\gamma(\theta)}{q_\psi(\theta)} \right] \quad (35)\\
&= \ln \frac{1}{S} \sum_{s=1}^{S} \mathbb{E}_{q_\psi(\theta^{(s)})}\left[\frac{p_\gamma(y|\theta^{(s)}, X) p_\gamma(\theta^{(s)})}{q_\psi(\theta^{(s)})} \right] \quad (36)\\
&= \ln \mathbb{E}_{\prod_{s=1}^{S} q_\psi(\theta^{(s)})}\left[\frac{1}{S} \sum_{s=1}^{S} \frac{p_\gamma(y|\theta^{(s)}, X) p_\gamma(\theta^{(s)})}{q_\psi(\theta^{(s)})} \right] \quad (37)\\
&\geq \mathbb{E}_{\prod_{s=1}^{S} q_\psi(\theta^{(s)})}\left[\ln \frac{1}{S} \sum_{s=1}^{S} \frac{p_\gamma(y|\theta^{(s)}, X) p_\gamma(\theta^{(s)})}{q_\psi(\theta^{(s)})} \right] =: \mathrm{ELBO}_S(\gamma, \psi). \quad (38)
\end{aligned}
$$

In Equation (36), the expectation in Equation (35) is replicated S times by introducing S i.i.d. variables $\theta^{(s)}$ and computing the average over those. In Equation (37), the expectation over $\theta^{(s)}$ is swapped with the sum before applying Jensen in Equation (38). The final importance-weighted

ELBO is denoted as $\text{ELBO}_S(\gamma, \psi)$ with an explicit dependence on the number of replicates S and where importance weights are given by the fraction between $p_\gamma(\theta^{(s)})$ and $q_\psi(\theta^{(s)})$.

It is straightforward to verify that the ordinary ELBO from Equation (29) in the previous section is recovered as a special case of the importance-weighted ELBO_S for $S = 1$. It turns out that the other extreme, when $S \to \infty$, recovers the log marginal likelihood, demonstrated as follows:

$$
\begin{aligned}
\lim_{S \to \infty} \text{ELBO}_S(\gamma, \psi) &= \lim_{S \to \infty} \mathbb{E}_{\prod_{s=1}^S q_\psi(\theta^{(s)})} \left[\ln \frac{1}{S} \sum_{s=1}^S \frac{p_\gamma(y|\theta^{(s)}, X)p_\gamma(\theta^{(s)})}{q_\psi(\theta^{(s)})} \right] \quad (39) \\
&= \mathbb{E}_{\prod_{s=1}^S q_\psi(\theta^{(s)})} \left[\ln \int q_\psi(\theta) \frac{p_\gamma(y|\theta, X)p_\gamma(\theta)}{q_\psi(\theta)} \, \mathrm{d}\theta \right] \quad (40) \\
&= \ln \int \cancel{q_\psi(\theta)} \frac{p_\gamma(y|\theta, X)p_\gamma(\theta)}{\cancel{q_\psi(\theta)}} \, \mathrm{d}\theta = \ln p_\gamma(y|X). \quad (41)
\end{aligned}
$$

It can be furthermore shown that the following sequence of inequalities holds in accordance with Burda et al. [2016] and Domke and Sheldon [2018]:

$$
\text{ELBO}(\gamma, \psi) = \text{ELBO}_1(\gamma, \psi) \leq \text{ELBO}_2(\gamma, \psi) \leq ... \leq \lim_{S \to \infty} \text{ELBO}_S(\gamma, \psi) = \ln p_\gamma(y|X), \quad (42)
$$

where the computational complexity is determined by the number of replicates S and increases from left to right. In the limit of infinite computational resources, $\ln p_\gamma(y|X)$ is recovered exactly. Note that ELBO_S is not only a tighter bound for large S, but also empirical estimates of ELBO_S (via sampling the outer expectation from $\theta^{(1)}$ to $\theta^{(S)}$) become more accurate and have less variance as S increases. This becomes apparent in the limit of $S \to \infty$ when every sample of the expectation over $\theta^{(1)}$ up to $\theta^{(S)}$ yields the same result, which is the exact log marginal likelihood.

For the sake of completeness, we provide here the importance-weighted formulation in case of a dataset with i.i.d. training samples $\{(y_n, X_n)\}_{n=1,..,N}$ by adjusting the likelihood accordingly:

$$
\text{ELBO}_S(\gamma, \psi) = \mathbb{E}_{\prod_{s=1}^S q_\psi(\theta^{(s)})} \left[\ln \frac{1}{S} \sum_{s=1}^S \frac{\prod_{n=1}^N p_\gamma(y_n|\theta^{(s)}, X_n)p_\gamma(\theta^{(s)})}{q_\psi(\theta^{(s)})} \right], \quad (43)
$$

that can be approximated with samples $\theta^{(s)}$ from each of the replicated distributions $q_\psi(\theta^{(s)})$, just like the non-i.i.d. formulation. Note that the way prediction is performed for new samples $X^\star$ is the same for importance-weighted VI as for vanilla VI, and Equations (30) and (32) from the previous section apply (for both the general and the i.i.d. case respectively).

3.3 Latent-Variable Variational Inference

The idea behind latent-variable VI is to introduce another latent variable h in addition to θ as illustrated by the graphical model in Figure 3 B) on the left ("general formulation"). The reason for this is to construct generative models that are more flexible as discussed shortly. To that end, it is assumed that the prior over θ and h factorizes into $p_\gamma(\theta)$ and $p_\gamma(h)$, and the likelihood $p_\gamma(y|\theta, h, X)$ is conditioned on h in addition to θ and X. We again indicate with γ the entirety of all generative parameters for notational convenience. Since h is latent, we need to do joint inference over θ and h. Under the typical assumption of a factorized approximate posterior $q_\psi(\theta)q_\psi(h)$, where ψ indicates all variational parameters, we arrive at the latent-variable ELBO:

$$
\begin{aligned}
\text{ELBO}(\gamma, \psi) &= \int \int q_\psi(\theta)q_\psi(h) \ln p_\gamma(y|\theta, h, X) \, \mathrm{d}h \, \mathrm{d}\theta \\
&\quad - \text{KL}\Big(q_\psi(h)\Big|\Big|p_\gamma(h)\Big) - \text{KL}\Big(q_\psi(\theta)\Big|\Big|p_\gamma(\theta)\Big), \quad (44)
\end{aligned}
$$

where the two KL terms are a result of the assumed factorization between θ and h in both the prior and the approximate posterior. Predicting $y^\star$ for previously unseen $X^\star$ is then achieved via:

$$
p(y^\star|X^\star) = \int \int p_\gamma(y^\star|\theta, h, X^\star)q_\psi(h) \, \mathrm{d}h \, q_\psi(\theta) \, \mathrm{d}\theta. \quad (45)
$$

At this point, one might wonder why we have introduced the latent variable h in the first place as it seems notationally redundant to θ. We shed light into this by providing a more concrete example

for the functional form of the likelihood. To that end, imagine the vanilla homoscedastic Bayesian neural net example from Section 3.1 where boldface $\boldsymbol{\theta}$ represents the vectorized weights of a neural net with a mean field multivariate prior $p(\boldsymbol{\theta})$. Under the latent-variable formulation, we need to introduce another prior over h and the homoscedastic Gaussian likelihood needs to be conditioned on h as well. This is where the difference between $\boldsymbol{\theta}$ and h becomes apparent: the mean is then defined as $\mu_{\text{lik}}(h, X) = f_{\boldsymbol{\theta}}(h, X)$ where $\boldsymbol{\theta}$ parameterizes the mean function as a neural net (indicated by the subscript $\boldsymbol{\theta}$) but h serves as additional neural network input. Ordinarily, without an additional latent variable h, the distribution over y for a given $\boldsymbol{\theta}$ and X is a unimodal Gaussian. However, by introducing h and adding it to the neural net input, the distribution over y for a given $\boldsymbol{\theta}$ and X becomes non-Gaussian (when integrated over h) and can assume multiple modes. A multimodal distribution is more expressive in the sense that it can model more challenging relationships between labels y and the corresponding inputs X.

The latent-variable formulation is typically combined with the assumption of an i.i.d. training dataset $\{(y_n, X_n)\}_{n=1,..,N}$—see the graphical model in Figure 3 B) on the right ("i.i.d. dataset"). Since the latent variable h is considered as additional likelihood input in addition to X, it also assumed i.i.d. across training examples and gets an index n. Under a factorized likelihood, the ELBO becomes:

$$
\begin{aligned}
\text{ELBO}(\gamma, \psi) \quad = \quad & \sum_{n=1}^{N} \int \int q_{\psi}(\theta) q_{\psi}(h_n) \ln p_{\gamma}(y_n | \theta, h_n, X_n) \, \mathrm{d}h_n \, \mathrm{d}\theta \\
& - \sum_{n=1}^{N} \text{KL}\Big(q_{\psi}(h_n) \Big| \Big| p_{\gamma}(h_n)\Big) - \text{KL}\Big(q_{\psi}(\theta)\Big|\Big| p_{\gamma}(\theta)\Big),
\end{aligned}
\tag{46}
$$

with a separate integral and KL term for each h_n. Predictions $\{y_n^{\star}\}_{i=1,..,N^{\star}}$ for new data points $\{X_n^{\star}\}_{n=1,..,N^{\star}}$ are then accomplished via the following formulation:

$$
p(y_1^{\star}, ..., y_{N^{\star}}^{\star} | X_1^{\star}, ..., X_{N^{\star}}^{\star}) = \int \prod_{n=1}^{N^{\star}} \int p_{\gamma}(y_n^{\star} | \theta, h_n, X_n^{\star}) p_{\gamma}(h_n) \, \mathrm{d}h_n \, q_{\psi}(\theta) \, \mathrm{d}\theta,
\tag{47}
$$

where, importantly, h_n is integrated out with the prior $p_{\gamma}(h_n)$ instead of with the approximate posterior, as opposed to the non i.i.d. formulation from Equation (45). The reason for integrating h_n with the prior is that, naively, in the course of training, there is a separate approximate posterior $q_{\psi}(h_n)$ for each individual training example (X_n, y_n). The approximate posterior $q_{\psi}(h_n)$ can therefore be interpreted as auxiliary training tool that does not readily generalize to unseen $X_n^{\star}$, and is typically "thrown away" after the training phase because it is no longer needed for prediction.

For illuminating purposes, let's provide a more concrete example for an i.i.d. regression problem with one-dimensional labels y_n. Imagine, similarly to earlier, that the prior $p(\boldsymbol{\theta})$ is a mean field multivariate Gaussian over vectorized weights $\boldsymbol{\theta}$ of a neural net. Let's furthermore imagine that $p(\mathbf{h}_n)$ is a multivariate standard normal Gaussian. Let the likelihood $p_{\gamma}(y_n|\boldsymbol{\theta}, \mathbf{h}_n, X_n)$ be a homoscedastic Gaussian with variance $\upsilon_{\text{lik}}^{(\gamma)}$—where γ indicates that the variance is a generative parameter—and a neural net mean function $\mu_{\text{lik}}(\mathbf{h}_n, X_n) = f_{\boldsymbol{\theta}}(\mathbf{h}_n, X_n)$ that has as input both $\mathbf{h}_n$ and X_n. Let the variational approximation $q_{\psi}(\boldsymbol{\theta})$ be a mean field multivariate Gaussian, and importantly, let's assume a multivariate Gaussian approximate posterior over $\mathbf{h}_n$ for every data point n denoted as $q_{\psi}(\mathbf{h}_n)$—where the variational parameters are mean-covariance-pairs for each n. We could alternatively parameterize the approximate posterior over $\mathbf{h}_n$ differently, e.g. as $q_{\psi}(\mathbf{h}_n|y_n, X_n)$ via a neural net that maps a (y_n, X_n)-tuple to a mean vector and covariance matrix for $\mathbf{h}_n$ (in which case the variational parameters were the weights of the neural net mapping, rather than an individual mean-covariance-pair for each training example n). The latter is called "amortized VI" and the variational neural net referred to as "recognition model" or "encoder". In this context, the mean function of the likelihood $\mu_{\text{lik}}(\mathbf{h}_n, X_n) = f_{\boldsymbol{\theta}}(\mathbf{h}_n, X_n)$, that is part of the generative model, is called the "decoder".

The example from the previous paragraph might sound familiar to some readers and provides indeed a conceptual generalization of a conditional variational autoencoder [Kingma et al., 2015, Sohn et al., 2015]. In a vanilla conditional variational autoencoder however, the setting is slightly simplified: the decoder—i.e. the neural net $\boldsymbol{\theta}$ parameterizing the likelihood mean $\mu_{\text{lik}}(\mathbf{h}_n, X_n) = f_{\boldsymbol{\theta}}(\mathbf{h}_n, X_n)$—is treated as a generative parameter γ (for which the optimization procedure finds a point estimate) rather than a latent variable over which one seeks to do inference. Coming back to the more general formulation where one seeks to do inference over $\boldsymbol{\theta}$. If we replace the neural net

weight prior $p(\boldsymbol{\theta})$ and the approximate posterior $q_\psi(\boldsymbol{\theta})$ with GPs that operate on the concatenated domain of X_n and $\mathbf{h}_n$, and replace $f_{\boldsymbol{\theta}}(\mathbf{h}_n, X_n)$ with $f(\mathbf{h}_n, X_n)$ accordingly where $f(\cdot)$ denotes a GP random function, we would obtain the GP equivalent of the example from the previous paragraph. The latter is similar to the work of Dutordoir et al. [2018].

Note that if we parameterize the approximate posterior over $\mathbf{h}_n$ as $q_\psi(\mathbf{h}_n|X_n)$ with a neural net that maps from X_n only (ignoring y_n) to a mean vector and covariance matrix for $\mathbf{h}_n$, we would sacrifice the label information during training but could predict $y_n^\star$ for unseen $X_n^\star$ context-dependently:

$$p(y_1^\star, ..., y_{N^\star}^\star | X_1^\star, ..., X_{N^\star}^\star) = \int \prod_{n=1}^{N^\star} \int p_\gamma(y_n^\star|\theta, h_n, X_n^\star) q_\psi(h_n|X_n^\star) \, \mathrm{d}h_n \; q_\psi(\theta) \, \mathrm{d}\theta, \qquad (48)$$

where we could now make use of the approximate posterior $q_\psi(h_n|X_n^\star)$ that is conditioned on $X_n^\star$ as opposed to Equation (47) where we were forced to make use of the less informative prior $p_\gamma(h_n)$ instead. Also note that we have deliberately reverted our notation in Equation (48) back from boldface $\boldsymbol{\theta}$ and $\mathbf{h}_n$ to θ and h_n in order to be notationally consistent with Equation (47).

3.4 Importance-Weighted Latent-Variable Variational Inference

Following Equation (38), we can straightforwardly go ahead and combine the idea of latent variables with the importance weighting trick in order to arrive at a tighter lower bound to the ELBO (that has less estimation variance):

$$\mathrm{ELBO}_S(\gamma, \psi) = \mathbb{E}_{\prod_{s=1}^S q_\psi(\theta^{(s)}) q_\psi(h^{(s)})} \left[\ln \frac{1}{S} \sum_{s=1}^S \frac{p_\gamma(y|\theta^{(s)}, h^{(s)}, X) p_\gamma(\theta^{(s)}) p_\gamma(h^{(s)})}{q_\psi(\theta^{(s)}) q_\psi(h^{(s)})} \right], \qquad (49)$$

from which we could obtain the corresponding formulation for an i.i.d. dataset $\{(y_n, X_n)\}_{n=1,..,N}$ by replacing the likelihood and the posterior approximation for h (as well as the prior for h) with their factorized counterparts.

However, there is an alternative way to combine importance weighting with the latent-variable formulation. Since h can be considered as additional likelihood input in addition to θ and X, as explained in the previous section, we can imagine the term $\int p(y|\theta, h, X) p_\gamma(h) \mathrm{d}h$ as actual likelihood of y given X and θ. We can then proceed with the ordinary VI formulation via an approximate posterior $q_\psi(\theta)$. This leaves us with a likelihood term that contains an integral over h, which we can lower-bound via importance weighting through approximate inference over h. The maths behind this idea is detailed as follows:

$$\ln p_\gamma(y|X) \quad = \quad \ln \int \int p(y|\theta, h, X) p_\gamma(h) \, \mathrm{d}h \; p_\gamma(\theta) \, \mathrm{d}\theta \qquad (50)$$

$$\geq \quad \int q_\psi(\theta) \ln \int p(y|\theta, h, X) p_\gamma(h) \, \mathrm{d}h \, \mathrm{d}\theta - \mathrm{KL}\Big(q_\psi(\theta) \Big|\Big| p_\gamma(\theta)\Big) \qquad (51)$$

$$\geq \quad \int q_\psi(\theta) \mathbb{E}_{\prod_{s=1}^S q_\psi(h^{(s)})} \left[\ln \frac{1}{S} \sum_{s=1}^S \frac{p(y|\theta, h^{(s)}, X) p_\gamma(h^{(s)})}{q_\psi(h^{(s)})} \right] \mathrm{d}\theta$$

$$- \mathrm{KL}\Big(q_\psi(\theta) \Big|\Big| p_\gamma(\theta)\Big), \qquad (52)$$

where in Equation (50), we applied Jensen's inequality at the θ level, and in Equation (51), we applied the importance-weighting trick at the level of the marginal term $\int p(y|\theta, h, X) p_\gamma(h) \mathrm{d}h$. This type of derivation encourages explicitly to counteract increased estimation variance of the ELBO as a consequence of introducing the additional latent variable h. We are going to come back to something similar in Section 4.3 in the context of VI with sparse latent-variable GPs.

We refrain at this stage from the formulation for an i.i.d. dataset that can be obtained readily. Also remember that the type of VI chosen (whether importance-weighted or vanilla VI) does not have an impact on how to predict new labels $y^\star$ given previously unseen data examples $X^\star$. Equations (45) and (47) from the previous section on vanilla latent-variable VI remain still valid for importance-weighted latent-variable VI expressions.

3.5   Bayesian Deep Learning and Bayesian Layers

It turns out that VI can be applied hierarchically with building blocks that are stacked on top of each other. To that end, imagine L random functions denoted in weight space view as $\{\theta^{(1)}, \theta^{(2)}, ..., \theta^{(L)}\}$ each of which is sampled from its own respective prior distribution $p_\gamma(\theta^{(l)})$ where γ indicates generative parameters and $l \in \{1, 2, ..., L\}$. The first random function $\theta^{(1)}$ receives as input the data point X and a sample from a latent variable $h^{(1)}$ distributed according to the prior $p_\gamma(h^{(1)})$, resulting in the random variable $f^{(1)}$. The second random function $\theta^{(2)}$ receives a sample from $f^{(1)}$ as input together with a sample from another latent variable $h^{(2)} \sim p_\gamma(h^{(2)})$, yielding $f^{(2)}$, and so forth. This process is repeated until the last layer L where the random variable $f^{(L)}$ is obtained. The random variable $f^{(L)}$ evaluates the likelihood for the label y as given by $p_\gamma(y|f^{(L)})$.

The graphical model behind the generative process of this hierarchical formulation is depicted in Figure 4 A) where the "likelihood layer" is highlighted through a red rectangle on the right. All the previous layers before the likelihood layer are building blocks with their own random function (that receives as input the output from the previous layer, as well as a block-specific latent random variable). These blocks are referred to as "Bayesian layers" [Tran et al., 2019], the second of which is highlighted through a red triangle in Figure 4 A) on the left. Note that the latent variables $h^{(l)}$ are optional for each layer and having one $h^{(l)}$ sitting at each layer l might be practically an overkill in terms of model flexibility, but we chose to represent the most general case.
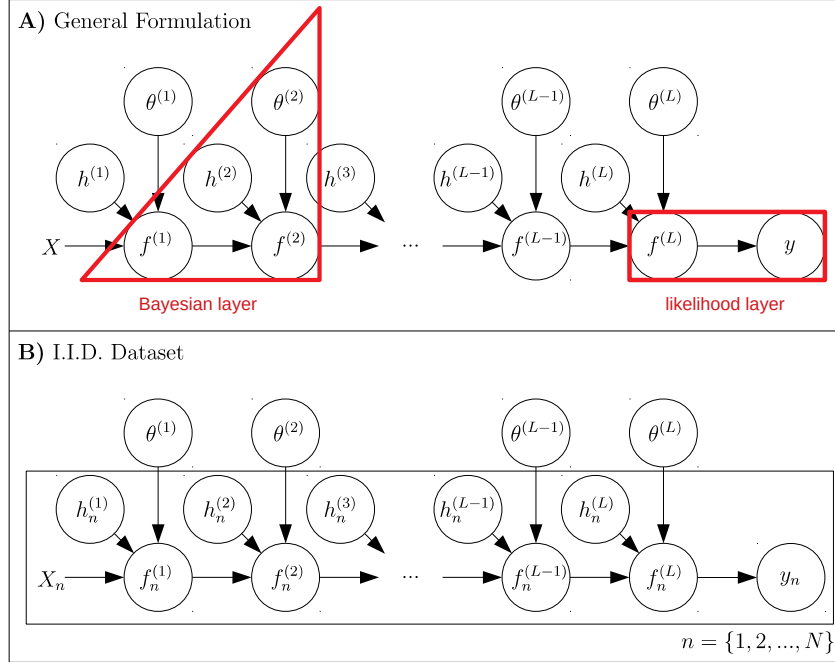

Figure 4: Graphical models for deep VI in supervised learning settings. X refers to inputs and y to labels. Random functions are depicted in weight space view through θ-notation (although we could have used alternatively the function space view in $f(\cdot)$-notation). The general formulation is depicted in **A)** and the formulation for an i.i.d. dataset in **B)** where an index n is added to variables that factorize over samples. The input X and the first latent variable $h^{(1)}$ determine where to evaluate the first random function $\theta^{(1)}$. This yields a random variable $f^{(1)}$ determining together with $h^{(2)}$ where to evaluate the second random function $\theta^{(2)}$, and so on. The last layer depicts the likelihood of y conditioned on $f^{(L)}$. The red triangle on the left indicates (the generative part of) a Bayesian layer. Bayesian layers are stacked on top of one another and serve as building blocks for deep VI. The triangle on the right indicates the likelihood layer on top of the last Bayesian layer. Bayesian layers also need to maintain an approximate posterior for approximate inference over their corresponding latent variables (i.e. $\theta^{(l)}$ as well as $h^{(l)}$ $\forall l$ in the general formulation, and $\theta^{(l)}$ as well as $h_n^{(l)}$ $\forall l$ and $\forall n$ in the i.i.d. formulation), as explained in the main text in more detail.

In order to perform inference over the latent functions θ and variables h, we need to introduce a posterior approximation for the joint distribution over all θ and h. A common way to model the posterior approximation is as pairwise independent in latent variables and functions, yielding $q_\psi(\theta^{(1)})q_\psi(h^{(1)})\cdots q_\psi(\theta^{(L)})q_\psi(h^{(L)})$ where ψ indicates variational parameters as earlier. Each $(q_\psi(\theta^{(l)}), q_\psi(h^{(l)}))$-pair is part of the corresponding Bayesian layer l. Under this assumption, the ELBO looks as follows:

$$\text{ELBO}(\gamma, \psi) = \int q_\psi(f^{(L)}) \ln p_\gamma(y|f^{(L)}) \, \mathrm{d}f^{(L)}$$
$$- \sum_{l=1}^{L} \text{KL}\Big(q_\psi(h^{(l)}) \Big|\Big| p_\gamma(h^{(l)})\Big) - \sum_{l=1}^{L} \text{KL}\Big(q_\psi(\theta^{(l)}) \Big|\Big| p_\gamma(\theta^{(l)})\Big), \quad (53)$$

where $q_\psi(f^{(L)})$ refers to the approximate posterior marginal over $f^{(L)}$ marginalized over all $\theta^{(l)}$ and $h^{(l)}$ for all layers l ranging from 1 to L, but also marginalized over all random variables $f^{(l)}$ for all layers ranging from 1 to $L-1$ (except the last of course). Note that each layer has its own respective latent-variable and latent-function KL term.

While propagating a single input X through the posterior approximations of the Bayesian layers (and the likelihood layer) in order to evaluate the expected log likelihood term in Equation (53), every layer needs to "keep track" of its contribution to the summed KL terms (that are part of the final ELBO objective). Furthermore, note that an end-to-end differentiable system is obtained via reparameterizing $\theta^{(l)}$, $h^{(l)}$ and $f^{(l)}$ at each layer [Kingma and Welling, 2014, Rezende et al., 2014]— the last $f^{(L)}$ then becomes a function of the variational parameters ψ of all the distributions $q_\psi(\theta^{(l)})$ and $q_\psi(h^{(l)})$ from $l = 1$ up to L. The expected log likelihood term in Equation (53) can then be readily approximated via samples obtained from randomly propagating X separately multiple times through all the layers.

Predicting new $y^\star$ for unseen $X^\star$ in the deep VI model from above is then accomplished via:

$$p(y^\star|X^\star) = \int p_\gamma(y^\star|f^{\star(L)})q_\psi(f^{\star(L)}) \, \mathrm{d}f^{\star(L)}, \quad (54)$$

where, similar to Equation (53), $q_\psi(f^{\star(L)})$ refers to the marginal distribution over $f^{\star(L)}$ as a result of marginalizing the approximate posterior over the latent variables $\theta^{(l)}$ and $h^{(l)}$ for all layers, as well as marginalizing over the latent variables $f^{\star(l)}$ for all layers except the last. The superscript star notation $\star$ refers to propagating new samples $X^\star$ through the approximate posterior model.

Figure 4 B) illustrates the graphical model for the generative process under an i.i.d. data assumption $\{(y_n, X_n)\}_{n=1,..,N}$ where latent variables $h_n^{(l)}$ at each layer l factorize over samples n but latent functions $\theta^{(l)}$ do not. Under an i.i.d. data assumption, the ELBO is similar to Equation (53):

$$\text{ELBO}(\gamma, \psi) = \sum_{n=1}^{N} \int q_\psi(f_n^{(L)}) \ln p_\gamma(y_n|f_n^{(L)}) \, \mathrm{d}f_n^{(L)}$$
$$- \sum_{n=1}^{N}\sum_{l=1}^{L} \text{KL}\Big(q_\psi(h_n^{(l)}) \Big|\Big| p_\gamma(h_n^{(l)})\Big) - \sum_{l=1}^{L} \text{KL}\Big(q_\psi(\theta^{(l)}) \Big|\Big| p_\gamma(\theta^{(l)})\Big), \quad (55)$$

except that there is a separate expected log likelihood term for each data point n, and a separate latent-variable KL term for $h_n^{(l)}$ at each layer l and data point n. There is however only one latent-function KL term for each function $\theta^{(l)}$ at each layer l because function parameters $\theta^{(l)}$ are the same for different data points n. Therefore, the latent-function KL terms are referred to as "global" whereas the the latent-variable KL terms as "local".

Predicting new $y_n^\star$ given new data points $X_n^\star$ requires to adjust Equation (54) accordingly, yielding:

$$p(y_1^\star, ..., y_{N^\star}^\star|X_1^\star, ..., X_{N^\star}^\star) = \int \prod_{n=1}^{N^\star} p_\gamma(y_n^\star|\mathbf{f}^{\star(L)})q_\psi(\mathbf{f}^{\star(L)}) \, \mathrm{d}\mathbf{f}^{\star(L)}, \quad (56)$$

by plugging in the i.i.d. likelihood and by computing the expectation with the approximate posterior marginal $q_\psi(\mathbf{f}^{\star(L)})$. Here, the notation $\mathbf{f}^{\star(L)}$ denotes the multivariate random variable at the last layer L that is obtained when jointly propagating all new evaluation points $X_n^\star$ through all the layers.

In order to keep the notation lightweight in this section, we used non-boldface symbols where possible, which does not mean that the corresponding variables need to be scalars. We also made mostly use of the weight space view that we thought most readers are more familiar with. We are going to revert back to function space view in the next section when addressing VI with sparse GPs, where we combine the contents from Section 2 with the contents from Section 3. This will provide an overview over contemporary GP techniques for principled and flexible approximate inference in multilabel problems, that e.g. enables usage of non-Gaussian and heteroscedastic likelihoods, in both a memory and computationally efficient manner.

4 Variational Inference with Sparse GPs

Sparse GPs, as introduced in Section 2, can approximate intractable posterior processes via a predefined number of inducing points or features that control memory and computational complexity. Inducing points (and the parameters of the distributions over their corresponding inducing variables) are treated as optimization arguments of an ELBO objective maximized in the course of VI. In the following, we assume a supervised learning setting in accordance with previous parts of the tutorial (with a particular focus on an i.i.d. data scenario). In Section 4.1, we explain how to do vanilla VI with sparse GP models, including some tricks commonly applied in practice. In Section 4.2, we introduce latent variables to sparse GPs in order to increase their flexibility, which we extend in Section 4.3 by the importance-weighting trick. After that, in Section 4.4, we transition to VI with deep sparse GPs that can naturally handle functions which might be "not smooth enough" for shallow GP models. Deep sparse GPs are extended by latent variables in Section 4.5 and combined with importance weighting in Section 4.6. Finally, in Section 4.7, we compare different sparse GP models on a synthetic example to highlight the practical benefit of importance-weighted deep latent-variable models when it comes to tackling a non-smooth and multimodal regression problem.

4.1 Shallow Sparse GPs

Let's commence with Bayesian inference for supervised learning with an i.i.d. training data set $\{(y_n, X_n)\}_{n=1,...,N}$ of size N, where y_n refers to a real-valued label associated with the training example X_n. The likelihood then factorizes over examples n, and the probability of observing a single y_n given the corresponding X_n is denoted as $p_\gamma(y_n|f(\cdot), X_n) = p_\gamma(y_n|f(X_n))$ where $f(\cdot)$ refers to an unknown function that is evaluated at X_n and distributed according to some prior singleoutput GP $p_\gamma(f(\cdot))$. For convenience, we again assume that γ refers to all generative parameters and that $f(\cdot)$ is a real-valued function (although the subsequent formulations remain valid under multilabel problems, vector-valued functions and multioutput GPs). The exact posterior process over $f(\cdot)$ can then be expressed with Bayes' rule as:

$$p_\gamma(f(\cdot)|y_1,...,y_N, X_1,...,X_N) = \frac{\prod_{n=1}^{N} p_\gamma(y_n|f(X_n))p_\gamma(f(\cdot))}{\int \prod_{n=1}^{N} p_\gamma(y_n|f(X_n))p_\gamma(f(\cdot))\mathrm{d}f(\cdot)}, \tag{57}$$

which is, in the most general setting under arbitrary likelihoods, no longer guaranteed to be a GP. If the likelihood is Gaussian with $f(X_n)$ being the mean for a particular y_n, then the posterior process would be a GP in closed form [Rasmussen and Williams, 2006]. However, this closed-form solution requires to invert a matrix with rows and columns equal to the number of training examples, which is cubic in N. The posterior is hence computationally intractable for large N even though a closed-form solution exists.

By resorting to VI under an approximate posterior sparse GP as given by Equation (12) and in the following denoted as $q_{\psi,\gamma}(f(\cdot))$, one can readily handle non-Gaussian likelihoods and control both memory and computational complexity at the same time. We ask the reader at this point to not get confused about the subscripts for variational and generative parameters ψ and γ—this aspect is a bit subtle and different from ordinary VI formulations, and will be explained in more detail in a subsequent paragraph. The following expression is agnostic to how inducing features are chosen, i.e. it is valid for ordinary inducing points but also for interdomain features. Under the i.i.d. setting,

the ELBO for sparse GPs can be written as:

$$\text{ELBO}(\gamma, \psi) = \sum_{n=1}^{N} \int q_{\psi,\gamma}(f(\cdot)) \ln p_{\gamma}(y_n | f(\cdot), X_n) \, \mathrm{d}f(\cdot) - \text{KL}\Big(q_{\psi,\gamma}(f(\cdot)) \Big|\Big| p_{\gamma}(f(\cdot))\Big) \quad (58)$$

$$= \sum_{n=1}^{N} \int q_{\psi,\gamma}(f(X_n)) \ln p_{\gamma}(y_n | f(X_n)) \, \mathrm{d}f(X_n) - \text{KL}\Big(q_{\psi}(\mathbf{u}) \Big|\Big| p_{\psi,\gamma}(\mathbf{u})\Big), \quad (59)$$

where the variational parameters ψ refer to the mean and covariance of the multivariate Gaussian distribution $q_{\psi}(\mathbf{u})$ over inducing variables $\mathbf{u}$—see Equation (3)—, as well as to inducing point locations $Z_1, ..., Z_M$ (or parameters of inducing features if they contain optimizable parameters). The generative parameters γ typically comprise hyperparameters of the prior kernel and the likelihood (e.g. the likelihood variance in case of a homoscedastic Gaussian likelihood). The term $p_{\psi,\gamma}(\mathbf{u})$ refers to the prior distribution over inducing variables $\mathbf{u}$ induced by the prior process $p_{\gamma}(f(\cdot))$. In the ordinary inducing point formulation, the inducing variables prior $p_{\psi,\gamma}(\mathbf{u})$ is simply the result of evaluating the prior GP at the inducing points Z_m. This explains the subscript ψ in $p_{\psi,\gamma}(\mathbf{u})$ because $p_{\psi,\gamma}(\mathbf{u})$ implicitly depends on inducing point locations that are variational parameters.

We will shortly explain how to go from Equation (58) to Equation (59), but before that, highlight a peculiarity that is different from ordinary VI objectives regarding variational and generative parameters ψ and γ. The approximate posterior process $q_{\psi,\gamma}(f(\cdot)) = \int p_{\psi,\gamma}(f(\cdot)|\mathbf{u})q_{\psi}(\mathbf{u})\mathrm{d}\mathbf{u}$, that lead to the formulation of a sparse GP in Equation (12) in the first place, depends by definition not only on variational parameters ψ but also on generative parameters γ. This is because the term $p_{\psi,\gamma}(f(\cdot)|\mathbf{u})$ is the prior GP $p_{\gamma}(f(\cdot))$ conditioned on inducing variables $\mathbf{u}$, and the prior GP is part of the generative model. So parameters of the prior process, like kernel hyperparameters, impact the approximate posterior GP directly. Be also aware that the notation $p_{\psi,\gamma}(f(\cdot)|\mathbf{u})$ hides some "dependencies", e.g. in the inducing point formulation, the inducing variables $\mathbf{u}$ are "assigned to" inducing points Z_m which are variational parameters, hence explaining the subscript ψ in $p_{\psi,\gamma}(f(\cdot)|\mathbf{u})$.

In going from Equation (58) to Equation (59), we have replaced integrals over infinite-dimensional random functions with integrals over finite-dimensional random variables, explained as follows. The expected log likelihood term in Equation (59) is merely a result of marginalization over $f(\cdot)$ as a consequence of the i.i.d. setting and the functional form imposed on the likelihood: y_n is conditioned on $f(\cdot)$ evaluated at X_n, and does not depend on function values at evaluation points other than X_n.

The second KL term however requires a bit of explanation. It represents the KL divergence between $f(\cdot)$ under the approximate posterior $q_{\psi,\gamma}(f(\cdot))$ and under the prior $p_{\gamma}(f(\cdot))$, and is mathematically equivalent to the finite-dimensional KL between the variational distribution $q_{\psi}(\mathbf{u})$ over the inducing variables $\mathbf{u}$ and the distribution over $\mathbf{u}$ under the prior process $p_{\psi,\gamma}(\mathbf{u})$. The latter equivalence can be shown following Matthews et al. [2016]:

$$\text{KL}\Big(q_{\psi,\gamma}(f(\cdot)) \Big|\Big| p_{\gamma}(f(\cdot))\Big) = \int q_{\psi,\gamma}(f(\cdot)) \ln \frac{q_{\psi,\gamma}(f(\cdot))}{p_{\gamma}(f(\cdot))} \, \mathrm{d}f(\cdot) \quad (60)$$

$$= \int q_{\psi,\gamma}(f(\cdot)) \ln \frac{p_{\psi}(\mathbf{u}|f(\cdot))q_{\psi,\gamma}(f(\cdot))}{p_{\psi}(\mathbf{u}|f(\cdot))p_{\gamma}(f(\cdot))} \, \mathrm{d}f(\cdot) \quad (61)$$

$$= \int \int q_{\psi,\gamma}(f(\cdot), \mathbf{u}) \ln \frac{q_{\psi,\gamma}(f(\cdot), \mathbf{u})}{p_{\psi,\gamma}(f(\cdot), \mathbf{u})} \, \mathrm{d}\mathbf{u} \, \mathrm{d}f(\cdot) \quad (62)$$

$$= \int \int p_{\psi,\gamma}(f(\cdot)|\mathbf{u})q_{\psi}(\mathbf{u}) \ln \frac{\cancel{p_{\psi,\gamma}(f(\cdot)|\mathbf{u})}q_{\psi}(\mathbf{u})}{\cancel{p_{\psi,\gamma}(f(\cdot)|\mathbf{u})}p_{\psi,\gamma}(\mathbf{u})} \, \mathrm{d}\mathbf{u} \, \mathrm{d}f(\cdot) \quad (63)$$

$$= \int q_{\psi}(\mathbf{u}) \ln \frac{q_{\psi}(\mathbf{u})}{p_{\psi,\gamma}(\mathbf{u})} \, \mathrm{d}\mathbf{u} = \text{KL}\Big(q_{\psi}(\mathbf{u}) \Big|\Big| p_{\psi,\gamma}(\mathbf{u})\Big). \quad (64)$$

The crucial part in the above equations is to understand how to get from Equation (60) to (62) where we have introduced the joint distributions over $f(\cdot)$ and $\mathbf{u}$ under the approximate posterior $q_{\psi,\gamma}(f(\cdot), \mathbf{u})$ and the prior $p_{\psi,\gamma}(f(\cdot), \mathbf{u})$. This step looks unintuitive because of the way the integration over $\mathbf{u}$ is introduced. We start by amending the fraction inside the log of Equation (60) by the conditional distribution $p_{\psi}(\mathbf{u}|f(\cdot))$, which is a Dirac delta function that is induced by a linear transformation through the interdomain features ϕ_m (or Z_m in the ordinary inducing point formulation), hence the subscript ψ because the interdomain transformation contains variational parameters.

Remember that $\mathbf{u}$ is a function of $f(\cdot)$ and completely determined by $f(\cdot)$, and that the log of Equation (61) contains inside the fraction of the joint over $f(\cdot)$ and $\mathbf{u}$ under the approximate posterior and the prior (because the conditional $p_\psi(\mathbf{u}|f(\cdot))$ is the same for both). The integral over the inducing variables $\mathbf{u}$ can therefore be introduced in Equation (62) where the joint assigns all probability mass to one particular $\mathbf{u}$ for a given $f(\cdot)$. In Equation (63), we then express the joints over $f(\cdot)$ and $\mathbf{u}$ inside the log "the other way around" with the other conditional-marginal pairs. Because of the way the approximate posterior has been defined, the term $p_{\psi,\gamma}(f(\cdot)|\mathbf{u})$ cancels. Since the log then no longer depends on $f(\cdot)$, we can marginalize over $f(\cdot)$ yielding a finite-dimensional integral over $\mathbf{u}$.

At this point, it might be insightful to present the graphical model of a sparse GP as illustrated in Figure 5 A). We have deliberately omitted the graphical model so far because we feel it is not necessarily intuitive nor helpful to start with for educational purposes.
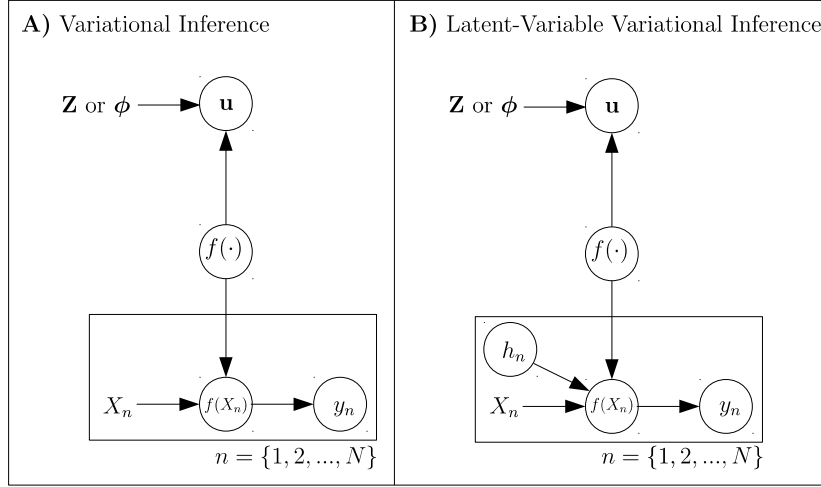

Figure 5: Graphical models for VI with sparse GPs in an i.i.d. supervised learning setting. X_n are training examples and y_n are labels. Both, random functions $f(\cdot)$ and inducing variables $\mathbf{u}$, are treated as latent variables. Evaluating a random function $f(\cdot)$ at a specific input location X_n yields the random variable $f(X_n)$. The latter determines the likelihood of the label y_n that corresponds to X_n. Inducing variables $\mathbf{u}$ are associated with inducing points $\mathbf{Z}$ (in the inducing point formulation) or with inducing features ϕ (in the interdomain formulation). The bold-face notation means that all M inducing points (or features) and their corresponding inducing variables are collapsed into one variable. This is necessary because the individual elements of the vector $\mathbf{u}$ are not pairwise independent. For the sake of memory and computational efficiency, the number of inducing points or features M is usually much smaller than the number of training examples N (indexed by n). The graphical model behind ordinary VI with sparse GPs is depicted in **A)** and subject of Section 4.1. The graphical model behind the corresponding latent-variable formulation, that requires another latent variable h_n, is depicted in **B)**—this is subject of Section 4.2. Also note that inducing points/features are not part of the generative model: given X_n, $\mathbf{u}$ is not required to generate y_n under the prior. The technical reason is that the node for $f(\cdot)$ has two outgoing arrows. Inducing points/features are merely part of the approximate posterior to infer $f(\cdot)$ and are hence variational parameters.

A practical advantage of the ELBO in Equation (59) is that the KL term has an analytical expression because of the Gaussian assumptions. In case of a Gaussian likelihood, the expected log likelihood term also enjoys a closed form expression. However, if the number of training data points N is too large, summing over all examples might be infeasible. One can then make use of minibatching in order to obtain an unbiased estimate of the expected log likelihood term [Hensman et al., 2013]. Under other likelihood models, for which no closed-form expressions exist, one has to resort to Monte Carlo methods or Gauss-Hermite quadrature [Hensman et al., 2015b]. In the latter case, the Gaussian random variable $f(X_n)$ is reparameterized [Kingma and Welling, 2014, Rezende et al., 2014] for optimization purposes. Also note that for singleoutput settings, where $f(X_n)$ is just a scalar, there is normally not much loss of accuracy incurred by Gauss-Hermite quadrature when approximating the expected log likelihood term.

Predictions $\{y_n^\star\}_{n=1,..,N^\star}$ for a new data set $\{X_n^\star\}_{n=1,..,N^\star}$ are straightforward with sparse GPs by multiplying the likelihood with the approximate posterior GP and integrating over function values:

$$p(y_1^\star, ..., y_{N^\star}^\star | X_1^\star, ..., X_{N^\star}^\star) = \int \prod_{n=1}^{N^\star} p_\gamma(y_n^\star | f(X_n^\star)) q_{\psi,\gamma}(\mathbf{f}^\star) \, \mathrm{d}\mathbf{f}^\star, \qquad (65)$$

where $q_{\psi,\gamma}(\mathbf{f}^\star)$ denotes the multivariate Gaussian obtained when evaluating the approximate posterior GP $q_{\psi,\gamma}(f(\cdot))$ at the new observations $X_1^\star, ..., X_{N^\star}^\star$. For a Gaussian likelihood, the predictive distribution has a closed form solution because it is Gaussian. For other likelihoods, one needs to resort once more to Monte Carlo methods or Gauss-Hermite quadrature.

At this point, the reader should have obtained a good understanding of VI with sparse GP models. We proceed with concluding this section by some parameterization tricks for inducing variables $\mathbf{u}$ that come in handy for practical applications. First, let's remind ourselves of how the variational distribution $q_\psi(\mathbf{u})$ over inducing variables is parameterized, namely as a multivariate Gaussian with a mean vector $\mathbf{m_u}$ and a covariance matrix $\mathbf{S_{uu}}$—see Equation (3) from the very beginning of this manuscript. The variable $\mathbf{m_u}$ enters the definition of a sparse GP in Equation (12) in terms of a difference $\mathbf{m_u} - \boldsymbol{\mu_u}$ in the mean function, where $\boldsymbol{\mu_u}$ is the mean of $\mathbf{u}$ under the prior. We can also define an alternative inducing variable $\tilde{\mathbf{u}} := \mathbf{u} - \boldsymbol{\mu_u}$ that is corrected under the prior mean $\boldsymbol{\mu_u}$, and parameterized via a mean vector $\mathbf{m_{\tilde{u}}}$ and a covariance matrix $\mathbf{S_{\tilde{u}\tilde{u}}}$, yielding the following:

$$f(\cdot) \sim \mathcal{GP}\Big(\mu(\cdot) + \mathbf{k_{\cdot u}} \mathbf{K_{uu}^{-1}} \mathbf{m_{\tilde{u}}}, k(\cdot, \cdot') - \mathbf{k_{\cdot u}} \mathbf{K_{uu}^{-1}} (\mathbf{K_{uu}} - \mathbf{S_{\tilde{u}\tilde{u}}}) \mathbf{K_{uu}^{-1}} \mathbf{k_{u \cdot'}} \Big). \qquad (66)$$

One might wonder what is the advantage of this alternative parameterization other than replacing $\mathbf{m_u} - \boldsymbol{\mu_u}$ with $\mathbf{m_{\tilde{u}}}$, and $\mathbf{S_{uu}}$ with $\mathbf{S_{\tilde{u}\tilde{u}}}$, in Equation (12)—there is no conceptual difference between $\mathbf{S_{uu}}$ and $\mathbf{S_{\tilde{u}\tilde{u}}}$ anyway. The answer is that the KL term in the ELBO from Equation (59) becomes more convenient to evaluate as $\mathrm{KL}\Big(q_\psi(\tilde{\mathbf{u}}) \big|\big| p_\gamma(\tilde{\mathbf{u}}) \Big)$. This is possible because the KL is invariant under said transformation of the random variable $\mathbf{u}$. Under the "transformed" KL, the distribution $p_\gamma(\tilde{\mathbf{u}})$ over $\tilde{\mathbf{u}}$ under the prior has zero mean with covariance matrix $\mathbf{K_{uu}}$. This is particularly helpful in the interdomain formulation introduced in Section 2.3, where computing $\boldsymbol{\mu_u}$ according to Equation (13) would require to integrate the product between the mean function $\mu(\cdot)$ and a feature function $\phi_m(\cdot)$ over the input domain $\mathcal{X}$. This integral does, in general, not have a closed-form expression for any mean function. Under the alternative parameterization with $\tilde{\mathbf{u}}$ however, such integration becomes unnecessary—using arbitrary mean functions is hence no longer a problem.

Exploiting the fact that the KL is invariant under "change-of-variables" transformations of the random variable $\mathbf{u}$ (or $\tilde{\mathbf{u}}$ respectively), we can go one step further and define $\hat{\mathbf{u}} := \mathrm{chol}(\mathbf{K_{uu}})\tilde{\mathbf{u}}$ with mean vector $\mathbf{m_{\hat{u}}}$ and covariance matrix $\mathbf{S_{\hat{u}\hat{u}}}$, and where chol refers to Cholesky decomposition. This yields $\mathrm{KL}\Big(q_\psi(\hat{\mathbf{u}}) \big|\big| p_\gamma(\hat{\mathbf{u}}) \Big)$ where $p_\gamma(\hat{\mathbf{u}})$ becomes the standard normal multivariate Gaussian (of dimension M). The latter parameterization requires in Equation (66) to replace $\mathbf{m_{\tilde{u}}}$ with $\mathrm{chol}(\mathbf{K_{uu}})^{-1} \mathbf{m_{\hat{u}}}$ and $\mathbf{S_{\tilde{u}\tilde{u}}}$ with $\mathrm{chol}(\mathbf{K_{uu}})^{-1} \mathbf{S_{\hat{u}\hat{u}}} \, \mathrm{chol}(\mathbf{K_{uu}})^{-\top}$. Note that this parameterization of $\hat{\mathbf{u}}$ is a specific form of whitening that can facilitate optimizing the ELBO from Equation (59) and is standard functionality provided in contemporary GP software frameworks [Matthews et al., 2017].

Another common trick is to initialize generative parameters (like kernel lengthscales for example) to "reasonable" values (if such values are known), and then optimize initially for the variational parameters only while keeping the generative parameters fixed. Once reasonable values for the variational parameters have been identified (e.g. after a fixed number of training iterations), generative parameters are "unclamped" and jointly optimized together with the variational parameters. Initializing generative parameters, like kernel hyperparameters, such that they are very far from optimal values can lead to large gradients that can impede the optimization process.

Finally, note that, as mentioned earlier, all formulations presented in this section remain valid under arbitrary likelihoods, multilabel problems, vector-valued functions and multioutput GPs. We can for example formulate a regression problem with a heteroscedastic Gaussian likelihood straightforwardly as follows [Saul et al., 2016]. Under vector-valued random functions $\mathbf{f}(\cdot)$ with two outputs, we can define a Gaussian likelihood as $p(y_n | \mathbf{f}(X_n))$ with mean $\mu_{\mathrm{lik}} = \mathbf{f}(X_n)[1]$ and variance $v_{\mathrm{lik}} = g(\mathbf{f}(X_n)[2])$, where 1 and 2 refer to both output indexes and where $g(\cdot)$ is a strictly positive function with real-line support (because GP function values are real-valued but variances need to be positive). This example is the conceptual function space equivalent to the neural network example presented earlier for VI under the weight space formulation in Section 3.1.

4.2 Latent-Variable Shallow Sparse GPs

According to Section 3.3, one can introduce another latent variable h_n for each training example n to make the generative model of a sparse GP more flexible. The corresponding graphical model is depicted in Figure 5 B). To that end, it is assumed that the prior over $f(\cdot)$ and all h_n factorizes into $p_\gamma(f(\cdot))$ and $\prod_{n=1}^N p_\gamma(h_n)$. The likelihood for a single example (X_n, y_n) is then defined as $p_\gamma(y_n|f(\cdot), h_n, X_n) = p_\gamma(y_n|f(h_n, X_n))$, where the notation $f(h_n, X_n)$ means that the GP operates on the concatenated domain of the input X_n and the latent variable h_n.

To understand why this increases modelling flexibility, imagine a homoscedastic Gaussian likelihood with mean $\mu_{\text{lik}} = f(h_n, X_n)$ and fixed variance. If there is no latent variable h_n, then the distribution over y_n for a given $f(\cdot)$ and X_n is unimodal. Under the latent-variable formulation however, the likelihood of y_n given $f(\cdot)$ and X_n can be multimodal (when integrated over h_n), hence becoming more expressive. When doing VI, we then need to perform inference over all h_n as well. The general assumption is that the approximate posterior factorizes as follows $q_{\psi,\gamma}(f(\cdot)) \prod_{n=1}^N q_\psi(h_n)$, where $q_{\psi,\gamma}(f(\cdot))$ depends on variational parameters ψ and generative parameters γ, as explained earlier, and $q_\psi(h_n)$ refers to the approximate posterior over h_n. The ELBO is then expressed as follows:

$$
\begin{aligned}
\text{ELBO}(\gamma, \psi) &= \sum_{n=1}^N \int \int q_{\psi,\gamma}(f(\cdot)) q_\psi(h_n) \ln p_\gamma(y_n|f(\cdot), h_n, X_n) \, \mathrm{d}h_n \, \mathrm{d}f(\cdot) \\
&\quad - \sum_{n=1}^N \text{KL}\Big(q_\psi(h_n) \Big| \Big| p_\gamma(h_n)\Big) - \text{KL}\Big(q_{\psi,\gamma}(f(\cdot)) \Big| \Big| p_\gamma(f(\cdot))\Big) \quad (67) \\
&= \sum_{n=1}^N \int q_\psi(h_n) \int q_{\psi,\gamma}(f(h_n, X_n)) \ln p_\gamma(y_n|f(h_n, X_n)) \, \mathrm{d}f(h_n, X_n) \, \mathrm{d}h_n \\
&\quad - \sum_{n=1}^N \text{KL}\Big(q_\psi(h_n) \Big| \Big| p_\gamma(h_n)\Big) - \text{KL}\Big(q_{\psi,\gamma}(\mathbf{u}) \Big| \Big| p_{\psi,\gamma}(\mathbf{u})\Big). \quad (68)
\end{aligned}
$$

The derivation is similar to VI without an additional latent variable as in Equation (59). Note that the integration over f and h_n has "swapped" between Equation (67) and (68) in the expected log likelihood term. Similar to earlier, one can integrate over $f(h_n, X_n)$ evaluated at specific locations (h_n, X_n), as opposed to all of $f(\cdot)$. The "swap" then occurs because h_n is part of the input to $f(\cdot)$. Also note that the innermost expectation over $f(h_n, X_n)$ in the first term of Equation (68) can be often computed efficiently (e.g. in closed form under a Gaussian likelihood).

The approximate posterior $q_\psi(h_n)$ can be chosen independently for each datapoint n, or in an amortized fashion via a parametric map $q_\psi(h_n|y_n, X_n)$ that maps training tuples (y_n, X_n) probabilistically to h_n. Conceptually, the model presented in this section is similar to the one presented in Dutordoir et al. [2018], and can be interpreted as function space equivalent of a generalized version of conditional variational autoencoders [Kingma et al., 2015, Sohn et al., 2015]. Conditional variational autoencoders naturally adopt the parameter space view but identify function parameters as point estimates rather than through inference.

While latent-variable sparse GPs are more flexible than ordinary sparse GPs, the ELBO in Equation (68) requires an additional sampling step over h_n. This additional sampling step leads to an increase in the variance of ELBO estimates. In the next section, we are going to present a method to alleviate this problem with the importance-weighting trick from earlier. But before that, let's finish this section with how to make predictions $\{y_n^\star\}_{n=1,..,N^\star}$ for new data examples $\{X_n^\star\}_{n=1,..,N^\star}$:

$$
p(y_1^\star, ..., y_{N^\star}^\star|X_1^\star, ..., X_{N^\star}^\star) = \int ... \int \int \prod_{n=1}^{N^\star} p_\gamma(y_n^\star|f(h_n^\star, X_n^\star)) p_\gamma(h_n^\star) q_{\psi,\gamma}(\mathbf{f}^\star) \, \mathrm{d}\mathbf{f}^\star \, \mathrm{d}h_{N^\star}^\star ... \, \mathrm{d}h_1^\star,
$$
(69)

where $q_{\psi,\gamma}(\mathbf{f}^\star)$ refers to the multivariate Gaussian that results from evaluating $q_{\psi,\gamma}(f(\cdot))$ at all $(h_n^\star, X_n^\star)$-pairs. Note that the integral over latent variables $h_n^\star$ is outermost because the multivariate random variable $\mathbf{f}^\star$ implicitly depends on all $h_n^\star$—since $f(\cdot)$ receives both $X_n^\star$ and $h_n^\star$ as joint input. Also remember that the prior $p_\gamma(h_n^\star)$ over latent variables $h_n^\star$ is required when predicting $y_n^\star$ for new data points $X_n^\star$. The latter is due to the fact that it is naively not possible to evaluate the approximate posterior over latent variables $h_n^\star$ for previously unseen datapoints $X_n^\star$—neither under the naive

formulation $q_\psi(h_n)$ where there is an approximate posterior for each training example X_n but none for unseen data points $X_n^\star$, nor under the amortized formulation where the approximate posterior $q_\psi(h_n|y_n, X_n)$ depends on labels that are unknown for novel data points at prediction time.

4.3 Importance-Weighted Latent-Variable Shallow Sparse GPs

As outlined in the previous section, one drawback of the latent-variable formulation is that another sampling step over h_n is required when compared to ordinary VI. This additional sampling step increases the variance of ELBO estimates which leads to decreased efficiency during optimization. We have seen in Section 3.4 how to exchange computational resources for a tighter lower bound to the ELBO that, at the same time, can be estimated more reliably with less estimation variance. The latter can be achieved with the importance-weighting trick as described earlier for the parameter space view in Equation (52) from Section 3.4. Applying the same kind of reasoning for shallow sparse GPs and i.i.d. training data yields in function space view:

$$
\begin{aligned}
\mathrm{ELBO}_S(\gamma, \psi) \;=\; & \sum_{n=1}^{N} \mathbb{E}_{\prod_{s=1}^{S} q_\psi(h_n^{(s)}) q_{\psi,\gamma}(\mathbf{f}_n)} \left[\ln \frac{1}{S} \sum_{s=1}^{S} \frac{p_\gamma(y_n|\mathbf{f}_n[s]) p_\gamma(h_n^{(s)})}{q_\psi(h_n^{(s)})} \right] \\
& - \mathrm{KL}\Big(q_\psi(\mathbf{u}) \Big\| p_{\psi,\gamma}(\mathbf{u}) \Big),
\end{aligned}
\tag{70}
$$

where S denotes the number of importance-weighted replicates for h_n indexed by s. The quantity $\mathbf{f}_n$ is the S-dimensional multivariate Gaussian random variable obtained when jointly evaluating the approximate posterior GP $q_{\psi,\gamma}(f(\cdot))$ at the locations $(h_n^{(s)}, X_n)$ for all S replications $h_n^{(s)}$ given X_n. Note that there is an implicit outer expectation in Equation (70) w.r.t. latent variables $h_n^{(1)}$ up to $h_n^{(S)}$ and an implicit inner expectation w.r.t. $\mathbf{f}_n$ that depends on latent variables from the outer expectation. The notation $\mathbf{f}_n[s]$ refers to the s-th component of the vector $\mathbf{f}_n$.

For sparse GPs, terms of the form $\int q_{\psi,\gamma}(f(h, X)) \ln p_\gamma(y|f(h, X)) \mathrm{d}f(h, X)$ are often efficiently computable—e.g. in closed form in case of a Gaussian likelihood. The problem with Equation (70) is that it does not contain such an expected log likelihood expression, as opposed to the naive ELBO formulation. In order to take advantage of efficient computability, it is therefore desirable to combine importance weighting with said expected log likelihood expression (following Salimbeni et al. [2019]). This can be achieved with a sequence of steps as detailed next. We start with introducing importance weights for latent variables h_n in the marginal likelihood:

$$
p_\gamma(y_1, ..., y_N|X_1, ..., X_N) = \mathbb{E}_{\prod_{n=1}^{N} p_\gamma(h_n)} \left[p_\gamma(y_1, ..., y_N|h_1, ..., h_N, X_1, ..., X_N) \right]
\tag{71}
$$

$$
= \mathbb{E}_{\prod_{n=1}^{N} q_\psi(h_n)} \left[p_\gamma(y_1, ..., y_N|h_1, ..., h_N, X_1, ..., X_N) \frac{\prod_{n=1}^{N} p_\gamma(h_n)}{\prod_{n=1}^{N} q_\psi(h_n)} \right],
\tag{72}
$$

where importance weights are introduced via an approximate posterior $\prod_{n=1}^{N} q_\psi(h_n)$ over all h_n. Note that $p_\gamma(y_1, ..., y_N|h_1, ..., h_N, X_1, ..., X_N)$ is the likelihood of all labels $\{y_1, ..., y_N\}$ conditioned on all latent variables $\{h_1, ..., h_N\}$ and training examples $\{X_1, ..., X_N\}$ but, importantly, marginalized over random functions $f(\cdot)$.

Next, we leverage the ordinary ELBO to lower-bound $\ln p_\gamma(y_1, ..., y_N|h_1, ..., h_N, X_1, ..., X_N)$ as:

$$
\begin{aligned}
\ln p_\gamma(y_1, ..., y_N|h_1, ..., h_N, X_1, ..., X_N) \;\geq\; & \sum_{n=1}^{N} \mathbb{E}_{q_{\psi,\gamma}(f(h_n, X_n))} \left[\ln p_\gamma(y_n|f(h_n, X_n)) \right] \\
& - \mathrm{KL}\Big(q_\psi(\mathbf{u}) \Big\| p_{\psi,\gamma}(\mathbf{u}) \Big).
\end{aligned}
\tag{73}
$$

Taking the exp of Equation (73) gives a lower bound to $p_\gamma(y_1, ..., y_N|h_1, ..., h_N, X_1, ..., X_N)$ which we can plug into Equation (72). After taking the log and some rearrangements, one arrives at:

$$
\begin{aligned}
\ln p_\gamma(y_1, ..., y_N|X_1, ..., X_N) \;\geq\; & \sum_{n=1}^{N} \ln \mathbb{E}_{q_\psi(h_n)} \left[\mathrm{explik}_{\gamma,\psi}(y_n, h_n, X_n) \frac{p_\gamma(h_n)}{q_\psi(h_n)} \right] \\
& - \mathrm{KL}\Big(q_\psi(\mathbf{u}) \Big\| p_{\psi,\gamma}(\mathbf{u}) \Big),
\end{aligned}
\tag{74}
$$

where, in order to preserve a clear view, we needed to define the helper term $\text{explik}_{\gamma,\psi}(y_n, h_n, X_n)$ for the exponential of the expected log likelihood term over function values $f(h_n, X_n)$:

$$\text{explik}_{\gamma,\psi}(y_n, h_n, X_n) := \exp\left(\int q_{\psi,\gamma}(f(h_n, X_n)) \ln p_\gamma(y_n|f(h_n, X_n)) \, \mathrm{d}f(h_n, X_n)\right). \quad (75)$$

The importance-weighting trick can finally be applied to the first log term on the r.h.s. of Equation (74) by introducing S replicates indexed by s for each h_n, as in Equation (70), yielding:

$$\ln p_\gamma(y_1, ..., y_N|X_1, ..., X_N) \geq \sum_{n=1}^{N} \mathbb{E}_{\prod_{s=1}^{S} q_\psi(h_n^{(s)})}\left[\ln \frac{1}{S} \sum_{s=1}^{S} \text{explik}_{\gamma,\psi}(y_n, h_n^{(s)}, X_n) \frac{p_\gamma(h_n^{(s)})}{q_\psi(h_n^{(s)})}\right]$$
$$- \text{KL}\left(q_\psi(\mathbf{u}) \middle\| p_{\psi,\gamma}(\mathbf{u})\right), \quad (76)$$

that combines importance weighting on the level of latent variables $h_n^{(s)}$ with the benefit of efficient computability of the expected log likelihood term over function values $f(h_n^{(s)}, X_n)$ inside the expression $\text{explik}_{\gamma,\psi}(y_n, h_n, X_n)$ as introduced in Equation (75). The next couple of sections deal with deep sparse GPs that are obtained by stacking shallow sparse GPs on top of each other, but we are going to return back to the trick of combining efficient computability and importance weighting in Section 4.6 later on after having introduced latent-variable deep sparse GPs.

4.4 Deep Sparse GPs

Following Section 2.5 and more precisely Figure 2, deep GPs hold the potential to model functions that are less smooth and more abruptly changing (with which an ordinary GP might have problems). In line with the conceptual logic of Bayesian deep learning from Section 3.5, one can create a generative model via stacking multioutput GPs on top of each other and feeding the final outcome into a likelihood. Imagine to that end L vector-valued random functions $\{\mathbf{f}^{(1)}(\cdot), ..., \mathbf{f}^{(L)}(\cdot)\}$ that are distributed according to L independent prior multioutput GPs with a joint distribution $p_\gamma(\mathbf{f}^{(1)}(\cdot)) \cdots p_\gamma(\mathbf{f}^{(L)}(\cdot))$. Note that the input domain of the first GP is $\mathcal{X}$ and the input dimension of each other GP equals the output dimension of the previous GP. Practically, we recommend to use an identity mean function in prior distributions (where possible, or linear otherwise) in all layers except the last. This is to encourage that different training inputs are mapped to different latent representations, which helps with the learning progress. Mapping different training inputs to the same latent representation on the other hand (as encouraged by a zero-mean function for example) could impede the learning progress.

As explained previously in Section 2.5, a single data point X_n is propagated through a deep GP as follows (see the second paragraph in Section 2.5 for multiple data points). A random function $\mathbf{f}^{(1)}(\cdot)$ from the prior GP $p_\gamma(\mathbf{f}^{(1)}(\cdot))$ is evaluated at X_n yielding the vector-valued random variable $\mathbf{f}_n^{(1)}$. A single sample from this random variable $\mathbf{f}_n^{(1)}$ serves as input to evaluate the second prior GP $p_\gamma(\mathbf{f}^{(2)}(\cdot))$ resulting in the random variable $\mathbf{f}_n^{(2)}$ (a single sample of which serves as input to the next prior GP, and so on). This ultimately yields the random variable $\mathbf{f}_n^{(L)}$, which determines the likelihood $p_\gamma(y_n|\mathbf{f}_n^{(L)})$ for the label y_n and the input X_n. Under an approximate posterior that factorizes over layers $l \in \{1, ..., L\}$, i.e. $q_{\psi,\gamma}(\mathbf{f}^{(1)}(\cdot)) \cdots q_{\psi,\gamma}(\mathbf{f}^{(L)}(\cdot))$ where each $q_{\psi,\gamma}(\mathbf{f}^{(l)}(\cdot))$ is a multioutput sparse GP, the ELBO is then defined as [Salimbeni and Deisenroth, 2017]:

$$\text{ELBO}(\gamma, \psi) = \sum_{n=1}^{N} \mathbb{E}_{q_{\psi,\gamma}(\mathbf{f}_n^{(L-1)})}\left[\int q_{\psi,\gamma}(\mathbf{f}_n^{(L)}|\mathbf{f}_n^{(L-1)}) \ln p_\gamma(y_n|\mathbf{f}_n^{(L)}) \, \mathrm{d}\mathbf{f}_n^{(L)}\right]$$
$$- \sum_{l=1}^{L} \text{KL}\left(q_\psi(\mathbf{U}^{(l)}) \middle\| p_{\psi,\gamma}(\mathbf{U}^{(l)})\right), \quad (77)$$

where inducing variables $\mathbf{U}^{(l)}$ are represented in matrix form in line with the output-as-output view. The first dimension of $\mathbf{U}^{(l)}$ corresponds to the number of inducing features (assuming the same number of inducing features per output for convenience) and the second dimension to the number of outputs at the layer l. It is practically recommended to initialize $q_\psi(\mathbf{U}^{(l)})$ with low variance in non-terminal layers to ensure a "deterministic" information flow during initial stages of training.

The quantity $q_{\psi,\gamma}(\mathbf{f}_n^{(L-1)})$ is the marginal distribution over $\mathbf{f}_n^{(L-1)}$, marginalized over all previous random variables from $\mathbf{f}_n^{(1)}$ up to $\mathbf{f}_n^{(L-2)}$. It is important to note that $q_{\psi,\gamma}(\mathbf{f}_n^{(L-1)})$ is not Gaussian and that there is no closed form expression for it—practically, one therefore needs to resort to Monte Carlo methods. On the other side, the term $q_{\psi,\gamma}(\mathbf{f}_n^{(L)}|\mathbf{f}_n^{(L-1)})$ is Gaussian by construction: it is the conditional probability of the last-layer random variable $\mathbf{f}_n^{(L)}$ conditioned on the the second-to-last random variable $\mathbf{f}_n^{(L-1)}$. The Gaussianity is simply due to the fact that $q_{\psi,\gamma}(\mathbf{f}_n^{(L)}|\mathbf{f}_n^{(L-1)})$ refers to the last approximate posterior GP $q_{\psi,\gamma}(f(\cdot))$ evaluated at $\mathbf{f}_n^{(L-1)}$. In ordinary non-deep notation, we usually omit conditioning explicitly on evaluation locations X_n because they are normally not considered as random variables. In a deep GP however, for all GPs except the first one, the inputs are random variables over which one needs to average.

We could have chosen to represent Equation (77) in terms of the marginal over $\mathbf{f}_n^{(L)}$ without explicitly mentioning $\mathbf{f}_n^{(L-1)}$ in the first place. However, because $q_{\psi,\gamma}(\mathbf{f}_n^{(L)}|\mathbf{f}_n^{(L-1)})$ is Gaussian, the inner expectation of the log likelihood in Equation (77) can often be computed efficiently, e.g. in case of a Gaussian likelihood. This would have been hidden if we had not emphasized the different expectations of the last and second-to-last layer. In order to obtain an end-to-end differentiable system, it is standard procedure to reparameterize the variables $\mathbf{f}^{(l)}$ at each layer l following [Kingma and Welling, 2014, Rezende et al., 2014] in accordance with earlier sections.

Predictions $\{y_n^\star\}_{n=1,..,N^\star}$ for new data points $\{X_n^\star\}_{n=1,..,N^\star}$ are then obtained as follows:

$$p(y_1^\star,...,y_{N^\star}^\star|X_1^\star,...,X_{N^\star}^\star) = \mathbb{E}_{q_{\psi,\gamma}(\mathbf{F}^{\star(L-1)})}\left[\int \prod_{n=1}^{N^\star} p_\gamma(y_n^\star|\mathbf{F}^{\star(L)}[n,:])q_{\psi,\gamma}(\mathbf{F}^{\star(L)}|\mathbf{F}^{\star(L-1)})\,\mathrm{d}\mathbf{F}^{\star(L)}\right],$$
(78)

where upper-case bold-face notation $\mathbf{F}^\star$ represents a matrix-valued random variable (which naturally arises when a multioutput GP is evaluated at multiple data points in the output-as-output view—see Section 2.5 for a detailed explanation). The first dimension of $\mathbf{F}^{\star(l)}$ corresponds to the number of evaluation points (here $N^\star$) and the second dimension to the number of output heads at the layer l. We again separate the expectation over $\mathbf{F}^{\star(L-1)}$ and $\mathbf{F}^{\star(L)}$ to highlight that the inner expected log likelihood term can often be computed efficiently. The notational form $\mathbf{F}^{\star(L)}[n,:]$ refers to the n-th row of the random variable $\mathbf{F}^{\star(L)}$ that sits in the last layer indexed with L. Note how this is different from the notation in Equation (77) where we used $\mathbf{f}_n^{(L)}$ with a subscript n instead. This is not arbitrary. In Equation (77), every data point n is propagated independently through the deep GP because of marginalization due to the functional form imposed on the likelihood. In Equation (78) however, all samples n need to be propagated jointly through the deep GP, as is also the case with predictions using shallow GP models presented earlier in Equation (65).

In the next section, we will introduce latent variables $h_n^{(l)}$ for each layer $l \in \{1,...,L\}$ to make deep sparse GP models more flexible. In this context, we are going to illustrate the graphical model behind the generative process in Figure 6. This graphical model also holds for the ordinary deep sparse GP model from this section if we ignore the nodes for latent variables $h_n^{(l)}$.

4.5 Latent-Variable Deep Sparse GPs

In line with earlier sections, one can increase the flexibility of a deep GP's modelling capabilities by introducing latent variables $h_n^{(l)}$ for each data point n and each layer l, distributed according to the joint prior $\prod_{n=1}^N p_\gamma(h_n^{(1)})\cdots p_\gamma(h_n^{(L)})$. The generative model is similar to the previous section, except that the input domain of each prior GP $p_\gamma(\mathbf{f}^{(l)}(\cdot))$ is concatenated with the domain where $h_n^{(l)}$ lives, explained in more detail as follows in the context of propagating a single example X_n through the deep GP. Evaluating the first prior GP $p_\gamma(\mathbf{f}^{(1)}(\cdot))$ requires to draw a sample from the the first latent variable $h_n^{(1)} \sim p_\gamma(h_n^{(1)})$ and concatenate it with X_n. The resultant multivariate random variable $\mathbf{f}_n^{(1)}$ determines together with $h_n^{(2)}$ where to evaluate the second prior GP $p_\gamma(\mathbf{f}^{(2)}(\cdot))$, and so on. The final random variable $\mathbf{f}_n^{(L)}$ eventually determines the likelihood $p_\gamma(y_n|\mathbf{f}_n^{(L)})$ of the label y_n associated with X_n. Under the assumption of a factorized approximate sparse posterior process $q_{\psi,\gamma}(\mathbf{f}^{(1)}(\cdot))\cdots q_{\psi,\gamma}(\mathbf{f}^{(L)}(\cdot))$, as in the previous section, and after introducing a factorized

approximate posterior for the additional latent variables $\prod_{n=1}^{N} q_\psi(h_n^{(1)}) \cdots q_\psi(h_n^{(L)})$, the ELBO can be expressed as [Salimbeni et al., 2019]:

$$
\begin{aligned}
\text{ELBO}(\gamma, \psi) \;=\; & \sum_{n=1}^{N} \mathbb{E}_{q_{\psi,\gamma}(\mathbf{f}_n^{(L-1)})} \left[\int q_\psi(h_n^{(L)}) \int q_{\psi,\gamma}(\mathbf{f}_n^{(L)}|h_n^{(L)}, \mathbf{f}_n^{(L-1)}) \ln p_\gamma(y_n|\mathbf{f}_n^{(L)}) \, \mathrm{d}\mathbf{f}_n^{(L)} \, \mathrm{d}h_n^{(L)} \right] \\
& - \sum_{n=1}^{N} \sum_{l=1}^{L} \text{KL}\Big(q_\psi(h_n^{(l)}) \Big\| p_\gamma(h_n^{(l)}) \Big) - \sum_{l=1}^{L} \text{KL}\Big(q_\psi(\mathbf{U}^{(l)}) \Big\| p_{\psi,\gamma}(\mathbf{U}^{(l)}) \Big),
\end{aligned} \tag{79}
$$

where there is an additional latent-variable KL term for each data point and each layer because of the factorization assumptions in both the prior and the approximate posterior over $h_n^{(l)}$. Note that $q_{\psi,\gamma}(\mathbf{f}_n^{(L-1)})$ refers to the marginal distribution over $\mathbf{f}_n^{(L-1)}$, marginalized over $\mathbf{f}_n^{(1)}$ up to $\mathbf{f}_n^{(L-2)}$ but also marginalized over $h_n^{(1)}$ up to $h_n^{(L-1)}$—which has no closed form and needs to be evaluated sampling-based, similarly to the previous section.

In Equation (79), we again express the innermost expected log likelihood averaged over $\mathbf{f}_n^{(L)}$ explicitly to highlight that this term can be computed efficiently under certain conditions—e.g. a Gaussian likelihood. The graphical model behind the generative process of a latent-variable deep sparse GP is depicted in Figure 6. It contains the generative model of an ordinary deep sparse GP without additional latent variables from the previous section as a special case when ignoring the nodes $h_n^{(l)}$.
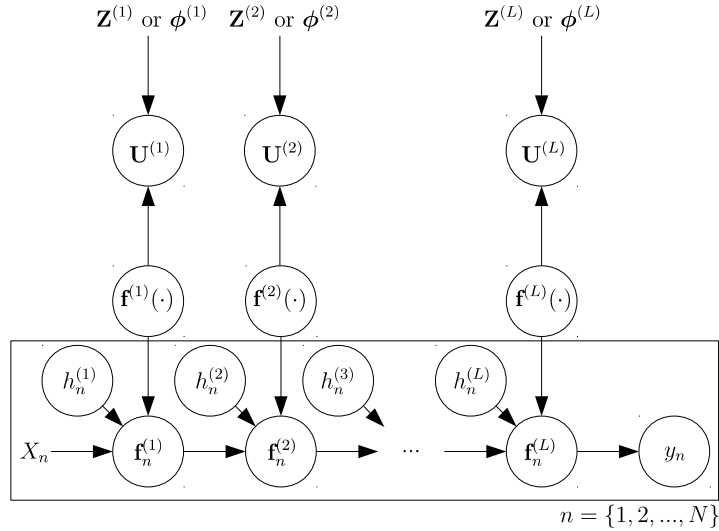

Figure 6: Graphical model of a latent-variable deep sparse GP in an i.i.d. supervised learning setting. Labels and data points are denoted as y_n and X_n respectively. An input X_n determines together with a sample from the first latent variable $h_n^{(1)}$ where to evaluate the first GP over vector-valued functions $\mathbf{f}^{(1)}(\cdot)$. This yields the random variable $\mathbf{f}_n^{(1)}$, a sample from which together with a sample from the second latent variable $h_n^{(2)}$ determines where to evaluate the second multioutput GP over $\mathbf{f}_n^{(2)}(\cdot)$, and so forth. A sample from the last random variable $\mathbf{f}_n^{(L)}$ in this cascade finally determines the probability of y_n. Inducing features are denoted as ϕ and contain inducing points $\mathbf{Z}$ as a special case (for Dirac features). Here, bold-face notation means that all M inducing features/points are stored in one variable for each layer l and shared across outputs for notational convenience. If we assume that inducing points/features are not shared across outputs, then each output would have its own set of M inducing points/features. Under the ordinary inducing point formulation for example, $\mathbf{Z}$ would then be a three-dimensional tensor ($M \times$ number of input dimensions $\times$ number of outputs) rather than a matrix ($M \times$ number of input dimensions). The associated inducing variables $\mathbf{U}$ are matrices where the first dimension corresponds to the number of inducing features M and the second dimension to the number of outputs (irrespective of whether inducing features are shared across outputs or not). The total number of inducing features M is usually much smaller than the number of training examples N for reasons of computational and memory efficiency.

Predicting new labels $\{y_n^\star\}_{n=1,..,N^\star}$ for new data $\{X_n^\star\}_{n=1,..,N^\star}$ is then similar to Equation (78):

$$p(y_1^\star,...,y_{N^\star}^\star|X_1^\star,...,X_{N^\star}^\star) = \mathbb{E}_{q_{\psi,\gamma}(\widetilde{\mathbf{F}^\star}^{(L-1)})}\left[\int \prod_{n=1}^{N^\star} p_\gamma(y_n^\star|\mathbf{F}^{\star(L)}[n,:])q_{\psi,\gamma}(\mathbf{F}^{\star(L)}|\widetilde{\mathbf{F}^\star}^{(L-1)})\,\mathrm{d}\mathbf{F}^{\star(L)}\right],$$

(80)

where the notation $\mathbf{F}^\star$ refers to matrix-valued random variables that result from jointly propagating all new data points $\{X_n^\star\}_{n=1,..,N^\star}$ through the deep GP, and where the short-hand notation $\widetilde{\mathbf{F}^\star}^{(L-1)} := (h_n^{(L)}, \mathbf{F}^{\star(L-1)})$ is introduced to preserve a clear view. At this stage, the reader should have obtained a good understanding of VI for deep sparse GP models with additional latent variables. What remains to be addressed is how to make latent-variable deep sparse GP models more optimization-efficient using the importance-weighting trick, as discussed next.

4.6 Importance-Weighted Latent-Variable Deep Sparse GPs

We have seen in Section 4.3 how to use the importance-weighting trick for efficient VI in shallow sparse GP models with an additional latent variable. The motivation behind this trick is to trade computational resources to obtain a tighter lower bound to the log marginal likelihood and to achieve less variance during optimization when computing ELBO estimates. The idea is to introduce importance weights in such a way that the innermost expected log likelihood term of the ELBO objective is preserved, since this term can often be computed efficiently.

In a deep sparse GP, the same logic applies to all latent variables from $h_n^{(1)}$ up to $h_n^{(L)}$ in accordance with Equation (76) from Section 4.3. A detailed derivation for a two-layer deep sparse GP with an additional latent variable in the first layer only (but not in the second) can be found in Salimbeni et al. [2019]. Building on top of the latter, we present here the more general case for L sparse GP layers. We stick with the setting of only one additional latent variable at the first layer (extending the formulation to an additional latent variable for each layer is conceptually straightforward but notationally cumbersome, and there is currently no empirical evidence that having more than one additional latent variable is beneficial). The derivation is a bit more subtle but similar to Section 4.3 and follows the same steps as the two-layer setting from Salimbeni et al. [2019]—the end result is:

$$\begin{aligned}\mathrm{ELBO}_S(\gamma,\psi) &= \sum_{n=1}^{N}\mathbb{E}_{\prod_{s=1}^{S}q_\psi(h_n^{(s)})q_{\psi,\gamma}(\mathbf{F}_n^{(L-1)}|\mathbf{h}_n)}\left[\ln\frac{1}{S}\sum_{s=1}^{S}\mathrm{explik}_{\gamma,\psi}(y_n,\mathbf{F}_n^{(L-1)}[s,:])\frac{p_\gamma(h_n^{(s)})}{q_\psi(h_n^{(s)})}\right]\\&\quad - \sum_{l=1}^{L}\mathrm{KL}\Big(q_\psi(\mathbf{U}^{(l)})\Big|\Big|p_{\psi,\gamma}(\mathbf{U}^{(l)})\Big),\end{aligned}$$

(81)

where the notation $h_n^{(s)}$ refers to the s-th replication of the additional latent variable h_n in the first layer for the data point n (and $\mathbf{h}_n$ is shorthand for the latent variable vector storing all replications for a given n). The matrix-valued random variable $\mathbf{F}_n^{(L-1)}$ (of shape $S\times$ number of outputs) denotes the output of the deep GP at the second-to-last layer when evaluated jointly at all S replications $(h_n^{(s)}, X_n)$. In order to preserve a clear view, $q_{\psi,\gamma}(\mathbf{F}_n^{(L-1)}|\mathbf{h}_n)$ refers to the marginal distribution over the output $\mathbf{F}_n^{(L-1)}$ at the second-to-last layer $L-1$ when marginalizing over all previous GP outputs, but conditioning on $\mathbf{h}_n$ that sits in the first layer. Similarly to earlier, and to ease the notation, we needed to introduce the helper function $\mathrm{explik}_{\gamma,\psi}(y_n,\mathbf{F}_n^{(L-1)}[s,:])$ defined as:

$$\mathrm{explik}_{\gamma,\psi}(y_n,\mathbf{F}_n^{(L-1)}[s,:]) := \exp\left(\int q_{\psi,\gamma}(\mathbf{f}_n^{(L,s)}|\mathbf{F}_n^{(L-1)}[s,:])\ln p_\gamma(y_n|\mathbf{f}_n^{(L,s)})\,\mathrm{d}\mathbf{f}_n^{(L,s)}\right), \quad (82)$$

where $\mathbf{f}_n^{(L,s)}$ refers to the output of the deep GP at the last layer L and $q_{\psi,\gamma}(\mathbf{f}_n^{(L,s)}|\mathbf{F}_n^{(L-1)}[s,:])$ to its conditional distribution conditioned on the output of the second-to-last layer $\mathbf{F}_n^{(L-1)}$ for a particular replication s. Note that the GP sitting at the last layer L can be evaluated for each individual replication s independently because of the factorized form of the likelihood, whereas all other latent layers require a joint evaluation across all S replications.

It can be insightful to continue at this point with a practical example that demonstrates the differences in modelling flexibility between shallow and deep sparse GPs. We do so in the next section where we also illustrate the effect of additional latent variables h_n on the model's expressiveness.
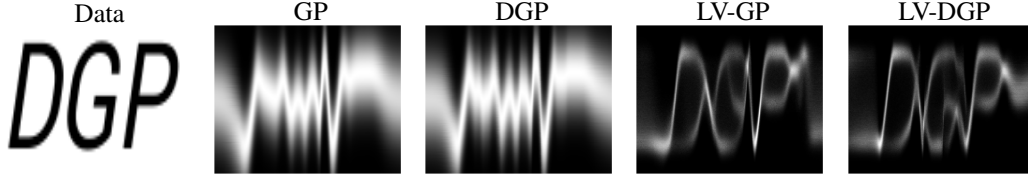
| Data | GP | DGP | LV-GP | LV-DGP |

Figure 7: Different GP models are compared to each other on a non-smooth multimodal regression problem following Salimbeni et al. [2019]. The first plot shows the training data set: data points (i.e. (X, y)-pairs) are given by the black pixels of the letters "*DGP*". The second plot shows predictive samples from a fitted shallow sparse GP model using a prior zero-mean function and a prior RBF kernel (in white on a black background). The third plot shows predictions from a deep GP consisting of two GPs stacked on top of each other (DGP). The fourth plot shows a shallow latent-variable GP (LV-GP). The last plot extends the two-layer deep GP from the third plot by adding a latent variable to the first layer (LV-DGP). The plots demonstrate the importance of latent variables in complex tasks: the quality of the fit increases from the left to the right achieving best results when combining deep GPs and latent variables.

4.7 Comparing Different Sparse GP Models on Synthetic Data

While the previous sections introduced the theory behind sparse GP models and extensions thereof, this section aims at providing an illuminating example to demonstrate the effect of deep and latent-variable GP models. To this end, we fit four different types of models on a non-smooth and multi-modal regression problem. The four models are

1. a shallow sparse GP according to Section 4.1,
2. a two-layer deep sparse GP according to Section 4.4,
3. a shallow sparse latent-variable GP according to Section 4.3,
4. and a two-layer deep sparse GP with a latent variable at the first layer (Section 4.6).

Note that the latent-variable models are trained with the importance-weighting trick as outlined earlier. The results are shown in Figure 7 and demonstrate the expressiveness obtained when combining deep GPs with latent variables (achieving the best result in this example). While the qualitative difference between shallow latent-variable and deep latent-variable sparse GPs might look minor in this example, the empirical results in Salimbeni et al. [2019] provide strong evidence of a significant performance boost across a wide range of regression problems when making latent-variable GPs deep. Also note that for a deep GP, latent variables could be added to any layer (not just to the first as in our example). However, we follow here Salimbeni et al. [2019] and leave it for future work to investigate whether adding more latent variables improves performance even further.

5 Further Reaching Topics

The following sections are an enumeration of VI objectives we deem important, but that would not have blended in nicely with the rest of this tutorial. In Section 5.1, we present latent-variable VI for unsupervised learning. In Section 5.2, we focus our attention on the non-sequential decision-making literature and how to recover the contextual bandits objective as an inference problem. Subsequently, in Section 5.3, we present a generalization of the former to the sequential decision-making setting that is typical for Markov decision processes and model-free reinforcement learning. Last, in Section 5.4, we address how to infer environment models that enable sequential decision-makers to plan virtually in the context of model-based reinforcement learning. This is extended to the partially observable setting in Section 5.5 where agents cannot observe the environment directly but need to rely on noisy observations instead.

5.1 Latent-Variable Variational Inference in Unsupervised Learning

In unsupervised learning, only observations y are available but no inputs X, and the goal is to infer parameters of a generative model that can mimic the true underlying distribution of the observations y. Throughout this tutorial, we have assumed a supervised learning scenario because it

contains the unsupervised setting (where input variables X are missing) as a special case. Deriving the vanilla ELBO from Section 3.1 for unsupervised learning is hence trivial, since all we need to do is to "delete" X in Equation (29). While the same is true for latent-variable VI, we nevertheless decided to dedicate a separate section to this topic at the end of this tutorial because of its importance in contemporary machine learning literature. The advantage of generative latent-variable models is that they can represent challenging distributions over y, e.g. containing multiple modes. The corresponding graphical model is, not surprisingly, similar to latent-variable VI for supervised learning from Section 3.3—compare Figure 8 A) to Figure 3 B) that only differ by the input variable X. The corresponding ELBO can be readily obtained from Equation (44) by "ignoring" X, yielding:

$$
\begin{aligned}
\text{ELBO}(\gamma, \psi) \quad = \quad & \int \int q_\psi(\theta) q_\psi(h) \ln p_\gamma(y|\theta, h) \, \mathrm{d}h \, \mathrm{d}\theta \\
& -\text{KL}\Big(q_\psi(h)\Big|\Big|p_\gamma(h)\Big) - \text{KL}\Big(q_\psi(\theta)\Big|\Big|p_\gamma(\theta)\Big),
\end{aligned}
\tag{83}
$$

where the likelihood term $p_\gamma(y|\theta, h)$ does not contain any other context variable apart from θ and h. Generating new samples $y^\star$ is then achieved via:

$$
p(y^\star) = \int \int p_\gamma(y^\star|\theta, h) q_\psi(h) \, \mathrm{d}h \, q_\psi(\theta) \, \mathrm{d}\theta.
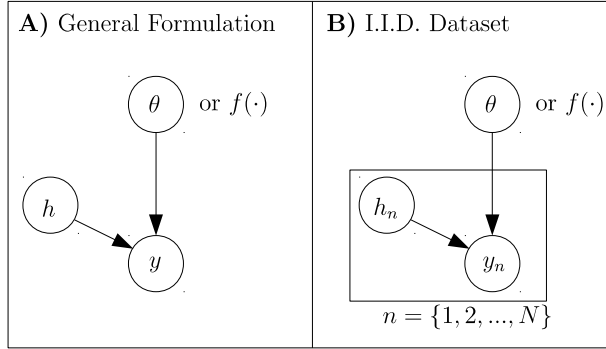\tag{84}
$$

Figure 8: Graphical models for latent-variable VI in unsupervised learning settings. Unknown functions are denoted as θ (in parameter space view) or $f(\cdot)$ (in function space view). They receive the additional latent variable h as input and map it probabilistically to the observed output y. **A)** refers to the ordinary setting and **B)** to the i.i.d. setting where N different examples are index by n.

Practically, one usually assumes i.i.d. data $\{y_n\}_{n=1,..,N}$ as indicated by the graphical model in Figure 8 B). In line with the supervised formulation, the additional latent variable h_n is also assumed i.i.d. across training examples and is indexed with n. The latter leads to a factorized likelihood which induces the following ELBO:

$$
\begin{aligned}
\text{ELBO}(\gamma, \psi) \quad = \quad & \sum_{n=1}^{N} \int \int q_\psi(\theta) q_\psi(h_n) \ln p_\gamma(y_n|\theta, h_n) \, \mathrm{d}h_n \, \mathrm{d}\theta \\
& - \sum_{n=1}^{N} \text{KL}\Big(q_\psi(h_n)\Big|\Big|p_\gamma(h_n)\Big) - \text{KL}\Big(q_\psi(\theta)\Big|\Big|p_\gamma(\theta)\Big),
\end{aligned}
\tag{85}
$$

Generating new observations $\{y_n^\star\}_{n=1,..,N^\star}$ is then accomplished via:

$$
p(y_1^\star, ..., y_{N^\star}^\star) = \int \prod_{n=1}^{N^\star} \int p_\gamma(y_n^\star|\theta, h_n) p_\gamma(h_n) \, \mathrm{d}h_n \, q_\psi(\theta) \, \mathrm{d}\theta,
\tag{86}
$$

where h_n needs to be integrated out with the prior $p_\gamma(h_n)$. This is in line with the supervised setting, where there is either a separate approximate posterior $q_\psi(h_n)$ for each individual training example y_n or an amortized approximate posterior $q_\psi(h_n|y_n)$ that maps observations y_n to latent variables h_n. Both parameterizations do not readily generalize to new observations $y_n^\star$ as they are

either implicitly or explicitly conditioned on training observations y_n. The approximate posterior $q_\psi(h_n)$ serves hence only as auxiliary training tool and is typically "thrown away" after learning.

Similar to earlier sections, it can be insightful to provide some concrete examples after presenting a general VI formulation. Imagine to that end an i.i.d. scenario where $\boldsymbol{\theta}$ represents the weights of a deep neural network in vectorized form, and where the prior $p(\boldsymbol{\theta})$ can be a multivariate Gaussian with diagonal covariance matrix. Imagine furthermore that $\mathbf{h}_n$ is a multivariate random variable as well, with a multivariate standard normal Gaussian prior $p(\mathbf{h}_n)$. Let the likelihood $p_\gamma(y_n|\boldsymbol{\theta}, \mathbf{h}_n)$ be homoscedastic Gaussian with variance $v_{\text{lik}}^{(\gamma)}$ and where γ indicates generative parameters as previously. Importantly, let the mean of the likelihood $\mu_{\text{lik}}(\mathbf{h}_n) = f_{\boldsymbol{\theta}}(\mathbf{h}_n)$ be a neural net function that receives $\mathbf{h}_n$ as input. We could then parameterize the approximate posterior over network weights $q_\psi(\boldsymbol{\theta})$ as a mean field multivariate Gaussian, and the approximate posterior over the additional latent variables $q_\psi(\mathbf{h}_n|y_n)$ in an amortized way via a neural net that maps y_n probabilistically to $\mathbf{h}_n$. In accordance with previous examples, ψ refers to the entirety of all variational parameters. We have just arrived at the general formulation of a variational auto-encoder according to the appendix of Kingma and Welling [2014], although using a homoscedastic rather than a heteroscedastic likelihood to ease the exposition. We can readily replace the neural net prior $p(\boldsymbol{\theta})$ and the corresponding approximate posterior $q_\psi(\boldsymbol{\theta})$ with GPs that operate on the domain where the additional latent variables $\mathbf{h}_n$ reside. This requires us to replace $f_{\boldsymbol{\theta}}(\mathbf{h}_n)$ with $f(\mathbf{h}_n)$ in the likelihood where $f(\cdot)$ is a GP random function. The latter leads us to a latent-variable GP model similar to the work of Damianou et al. [2016].

5.2   Contextual Bandits as Inference

Contextual bandits cover a class of non-sequential decision-making problems [Lattimore and Szepesvari, 2020]. Formally, there is a state or context variable $s \in \mathcal{S}$ and an action variable $a \in \mathcal{A}$, and the decision-maker's task is to take in each state s an optimal action a where optimality is defined through some real-valued reward function $R(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Clearly, the optimal strategy would be $\text{argmax}_a R(s,a)$ if the reward-maximizing argument can be identified for a given state s. Note that a finite action set $\mathcal{A}$ refers to the classical contextual bandit setting while a continuous action set describes the setting of contextual Bayesian optimization.

It turns out that this simple decision-making paradigm can be recovered under a soft policy constraint from an inference formulation, as explained in the following. For convenience, let's introduce the latent variable $\tau = (s,a)$ as short-hand notation for a state-action pair. Imagine a factorized prior for τ of the form $p(\tau) = p(s)\pi_0(a)$, where $p(s)$ is a prior distribution over states and $\pi_0(a)$ is a prior action distribution. Phrasing an inference problem requires then to specify an observed variable o and a likelihood that determines the probability of o conditioned on the latent variable τ. We make the artificial choice of $o \in \{0, 1\}$ being binary and define $p(o = 1|\tau) = \frac{\exp(R(s,a))}{\int \int \exp(R(s,a))\text{d}a\text{d}s}$. We furthermore artificially determine that only one observation o has been made and that this observation is 1. The corresponding graphical model plus a description of the prior and the likelihood is depicted in Figure 9 A) and B). Under an approximate posterior of the form $q(\tau) = p(s)\pi(a|s)$ that uses a state-conditioned action distribution $\pi(a|s)$, a.k.a. policy, we arrive at the following ELBO:

$$\text{ELBO}(\pi_0, \pi) = \int q(\tau) \ln p(o = 1|\tau) \, \text{d}\tau - \text{KL}\Big(q(\tau)\Big|\Big|p(\tau)\Big) \tag{87}$$

$$\equiv \int \int p(s)\pi(a|s)R(s,a) \, \text{d}a \, \text{d}s - \int p(s)\text{KL}\Big(\pi(a|s)\Big|\Big|\pi_0(a)\Big) \, \text{d}s, \tag{88}$$

where the equivalence sign "$\equiv$" in Equation (88) is due to the fact that the normalization constant in the denominator of $p(o = 1|\tau)$ does neither depend on the generative action prior π_0 nor on the variational policy π, and is hence dropped since it would just add an additional term that is unaffected by the optimization arguments (π_0, π).

It can be seen that the objective in Equation (88) recovers the original contextual bandit problem when not optimizing over the prior π_0 but only over the policy π, but importantly under a soft policy constraint that encourages the optimal policy to be close to the prior action distribution. In fact, one can weight the expected KL penalty in Equation (88) with a non-negative scalar β to control the trade-off between reward maximization and penalization—we touched on this approach before in Section 3.1 where it was referred to as β-VI. In β-VI, one recovers the contextual bandit objective precisely as $\beta \to 0$. The decision-making counterpart of β-VI is called "information-theoretic bounded rationality", see Ortega and Braun [2013] and Ortega et al. [2015] for details.

Going one step further, one can optimize over the generative action prior π_0 in addition to the variational policy π. It turns out that the optimization problem becomes then mathematically equivalent to rate distortion theory from information theory [Cover and Thomas, 2006]. In short, under the latter view, the decision-maker can be interpreted as information-theoretic channel with input s and output a, the reward function as some measure of quality for recovering the information of s in a, and the KL penalty as maximal information transmission rate. A detailed discussion of the topic is however beyond the scope of this tutorial and we refer the interested reader to Genewein et al. [2015]. Applications of the rate distortion framework for decision-making can be found in the literature across different research fields, e.g. economics [Sims, 2010], theoretical neuroscience [Leibfried and Braun, 2015] and machine learning [Leibfried and Braun, 2016]. Most recent advances aim at extending the framework to hierarchical decision-making settings, where lower-level decision-making routines pre-process the context yielding intermediate outcomes, that are then further processed by higher-level decision-making routines [Peng et al., 2017, Hihn et al., 2019, Hihn and Braun, 2020].

A) Graphical Model	**B)** Contextual Bandits	**C)** Reinforcement Learning
τ o	$\tau = s, a$ $p(\tau) = p(s)\pi_0(a)$ $p(o=1\|\tau) = \dfrac{\exp(R(s,a))}{\int\int \exp(R(s,a))\, \mathrm{d}a\, \mathrm{d}s}$	$\tau = s_0, a_0, s_1, a_1, ..., s_T, a_T$ $p(\tau) = p(s_0)\pi_0(a_0)P(s_1\|s_0,a_0)\pi_0(a_1)\cdots P(s_T\|s_{T-1},a_{T-1})\pi_0(a_T)$ $p(o=1\|\tau) = \dfrac{\exp(\sum_{t=0}^T R(s_t,a_t))}{\int\int \exp(\sum_{t=0}^T R(s_t,a_t))\, \mathrm{d}a_{0:T}\, \mathrm{d}s_{0:T}}$

Figure 9: Graphical models when interpreting decision-making as inference problems. **A)** shows a generic graphical model with a latent variable τ and an observed binary variable $o \in \{0,1\}$. Both of these variables have similar but different meanings in non-sequential decision-making settings of the contextual bandits type (as depicted in **B)**), and in sequential decision-making settings of the reinforcement learning type (as depicted in **C)**). In the contextual bandits setting (Section 5.2), τ refers to a state-action pair (s,a) and the probability of $o=1$ conditioned on τ is proportional to the $\exp$ of the reward $R(s,a)$ obtained when executing the action a in the state s. Here, $p(s)$ refers to the distribution over states and $\pi_0(a)$ to a prior action distribution. In the reinforcement learning setting (Section 5.3), τ refers to a state-action trajectory $(s_0, a_0, s_1, a_1, ..., s_T, a_T)$ within a Markov decision process, and the probability of $o=1$ conditioned on τ is proportional to the $\exp$ of the cumulative reward $\sum_{t=0}^T R(s_t,a_t)$. For the latter, we assume a finite time horizon where T indicates the end of the horizon. The notions $s_{0:T}$ and $a_{0:T}$ are short-hand notations for state and action trajectories respectively. $p(s_0)$ refers to the initial state distribution at time step $t=0$, $\pi_0(a_t)$ to a prior action distribution as in the non-sequential setting, and $P(s_{t+1}|s_t, a_t)$ refers to the state-transition probability distribution of transitioning to the state s_{t+1} at time step $t+1$ when executing the action a_t in the state s_t at time step t. Note that in both decision-making settings, the assumption is always that there is only one single observation for o which is 1—see text for details.

5.3 Reinforcement Learning as Inference

As opposed to contextual bandits, reinforcement learning [Sutton and Barto, 1998] covers sequential decision-making problems where an agent needs to identify a sequence of actions that maximizes cumulative reward. Mathematically speaking, the agent operates in an environment that is in a certain state $s_t \in \mathcal{S}$ at every time step t. The agent needs to choose an action $a_t \in \mathcal{A}$ after observing s_t. This yields an instantaneous reward $R(s_t, a_t) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as defined by a real-valued reward function (as in the contextual bandit case), and the environment transitions to a subsequent state s_{t+1}. Importantly, transitioning to a subsequent state is determined by a state-transition probability distribution $P(s_{t+1}|s_t, a_t)$ that is a conditional probability distribution over s_{t+1} conditioned on s_t and a_t. This problem definition describes a Markov decision process (MDP) and the agent's task is to maximize expected cumulative reward $\mathbb{E}\left[\sum_{t=0}^T R(s_t, a_t)\right]$ where T determines the time horizon. Here, the expectation is w.r.t. three components: the initial state distribution $p(s_0)$ that samples the first state at time step $t=0$, the state-transition distribution $P(s_{t+1}|s_t, a_t)$, and the agent's policy $\pi(a_t|s_t)$ which is a conditional probability distribution over a_t conditioned on s_t. The goal of the agent is hence to find an optimal policy $\pi(a_t|s_t)$ that yields highest expected cumulative reward.

34

The difficulty compared to the contextual bandit setting is that an action a_t at time step t not only affects the instantaneous reward at that time step, but also future rewards through future states. Note that we chose a finite-horizon problem with a stationary policy for the problem definition, because it will come in handy shortly when expressing the reinforcement learning problem as an inference problem. Also note that the contextual bandit formulation from the previous section can be recovered as a special case when defining the initial state distribution $p(s_0)$ to be the same as the state-transition distribution, i.e. $P(s_{t+1}|s_t, a_t) = p(s_{t+1})$ which does not depend on s_t and a_t.

Phrasing the reinforcement learning problem as an inference problem is conceptually similar to the contextual bandit scenario, but slightly more complex due to the sequential nature of the problem. In line with the previous section, let's introduce a latent variable $\tau = s_0, a_0, s_1, a_1, ..., s_T, a_T$ as short-hand notation for a state-action trajectory. Assume a factorized prior for τ of the following form: $p(\tau) = p(s_0)\pi_0(a_0)\prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi_0(a_{t+1})$ where $\pi_0(a_t)$ refers to a prior action distribution. We again introduce an artificial binary observed variable $o \in \{0, 1\}$ and define the likelihood as $p(o = 1|\tau) = \frac{\exp(\sum_{t=0}^{T} R(s_t, a_t))}{\int\int \exp(\sum_{t=0}^{T} R(s_t, a_t))\mathrm{d}a_{0:T}\mathrm{d}s_{0:T}}$ where $s_{0:T}$ and $a_{0:T}$ is short-hand for state- and action-trajectories respectively. As earlier, we assume that only one observation o has been made and that this observation is 1. This is summarized in Figure 9 A) and C). Using an approximate posterior of the form $q(\tau) = p(s_0)\pi(a_0|s_0)\prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_{t+1}|s_{t+1})$ with a state-conditioned variational policy $\pi(a_t|s_t)$, we obtain an ELBO expression as:

$$\mathrm{ELBO}(\pi_0, \pi) = \int q(\tau)\ln p(o = 1|\tau)\,\mathrm{d}\tau - \mathrm{KL}\Big(q(\tau)\big|\big|p(\tau)\Big) \tag{89}$$

$$\equiv \sum_{t=0}^{T}\int\int p_t^{(\pi)}(s_t)\pi(a_t|s_t)R(s_t, a_t)\,\mathrm{d}a_t\,\mathrm{d}s_t$$

$$-\sum_{t=0}^{T}\int p_t^{(\pi)}(s_t)\mathrm{KL}\Big(\pi(a_t|s_t)\big|\big|\pi_0(a_t)\Big)\,\mathrm{d}s_t, \tag{90}$$

where $p_t^{(\pi)}(s_t)$ refers to the marginal state distribution of the policy π at time step t, which is obtained by taking $q(\tau)$ and integrating out all trajectory elements except for the state at time step t. As in the previous section, the equivalence sign "$\equiv$" in Equation (90) is because the denominator in $p(o = 1|\tau)$ can be dropped since it just leads to an additive term that does not depend on π_0 and π.

It becomes apparent that Equation (90) recovers the original reinforcement learning objective but under a soft policy constraint that encourages the optimal policy to be close to a prior action distribution (when optimizing for π but keeping π_0 fixed). A more detailed derivation can be found in Levine [2018], but the underlying problem description remains also valid when using a state-conditioned action prior $\pi_0(a_t|s_t)$ rather than a state-unconditioned one. Similar to before, we can introduce a non-negative scaling factor β to weight the KL penalty at each time step. The latter can be referred to as "soft cumulative reward maximization" which has been described in infinite-horizon MDP settings [Azar et al., 2011, Rubin et al., 2012, Neu et al., 2017] where $T \to \infty$.

Under the special case of using a uniform prior action distribution $\pi_0(a_t) = \mathcal{U}(\mathcal{A})$, one arrives at the framework of entropy regularization for MDPs [Ziebart, 2010, Nachum et al., 2017]. Entropy regularization for MDPs has spurred quite a few reinforcement learning algorithms in the recent years that improved performance over ordinary non-regularized algorithms in a wide range of domains. These domains include grid worlds [Fox et al., 2016], robotics domains [Haarnoja et al., 2017, 2018, 2019], but also visual domains such as Atari games [Schulman et al., 2017, Leibfried et al., 2018]. In the meanwhile, the concept found also application in two-player scenarios where it is referred to as "tuneable artificial intelligence" meant to balance an artificial player's performance through the KL penalty such as to keep a human player engaged—see Grau-Moya et al. [2018] for details.

When optimizing over π_0 in addition to π, a generalization of the rate distortion framework for non-sequential decision-making (presented at the end of the previous section) is obtained. This was first described by Tishby and Polani [2011], and recent advances led to reinforcement learning algorithms that improved over entropy-regularized methods in Atari and robotics domains [Grau-Moya et al., 2019, Leibfried and Grau-Moya, 2019]. While the rate distortion framework fits into the inference perspective, recent research presents alternative information-theoretic formulations beyond rate distortion theory which enable more sophisticated prior action distributions [Tiomkin and Tishby, 2018, Leibfried et al., 2019] that can be conditioned on past or future observations.

We conclude this section by pointing out that KL regularization is a widely adopted technique in various fields of sequential decision-making. While enumerating all of them here would be beyond the scope of this manuscript, we want to point out a few in the further vicinity of soft cumulative reward maximization. Notable use of different types of KL regularization have been adopted in policy search [Peters et al., 2010, Schulman et al., 2015] and decision theory [Karny, 2020]. But also noteworthy is work in the field of robust MDPs [Grau-Moya et al., 2016] and multi-task learning [Petangoda et al., 2019] where KL regularization is applied on the level of environment models or latent-variable distributions respectively (in addition to the action level).

5.4 Variational Inference for Model-Based Reinforcement Learning

The reinforcement-learning-as-inference framework usually assumes a "model-free" perspective where an agent learns by trial and error, but does neither know nor try to infer an environment model. An environment model is useful for "model-based" reinforcement learning that enables an agent e.g. to virtually plan ahead in order to identify an optimal course of action. The prospect of model-based learning is to be more sample-efficient, i.e. to obtain good performance with less agent-environment interactions, but the drawback is that additional computational resources are required both for model learning but also for planning (which requires an internal virtual simulation). Since model-based learning has received increased attention in the recent years, we decided to dedicate a separate section to the subject in order to outline how to use VI to learn an environment model from agent-environment interaction data. Note that a tutorial on model-based reinforcement learning is beyond the scope of this manuscript, but we will provide some examples at the end of this section to provide an intuition for how models can be used for sequential decision-making.
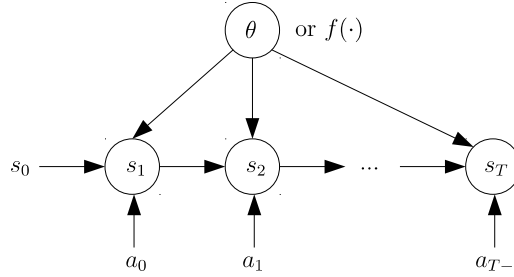

Figure 10: Graphical model for VI to infer the state-transition distribution for model-based reinforcement learning. Environment states are indicated with s_t and actions, taken by the agent, with a_t where $t \in \{0, 1, ..., T\}$ is a time index. The unknown state-transition distribution is denoted as θ in parameter space view (or as $f(\cdot)$ in function space view). Note that the inference procedure is conditioned on the initial state s_0 as well as on all actions from a_0 up to a_{T-1}.

Going ahead to phrase an inference problem, we need to specify a generative model first. For the sake of simplicity, we assume in the following that we are only interested in inferring the state-transition distribution, but not the initial state distribution or the reward function (although this would be possible with straightforward extensions). For this reason, our generative model is conditioned on the initial state and all actions. This leads us to the graphical model in Figure 10. The basic idea is that the state-transition distribution $P(s_{t+1}|s_t, a_t)$ from the previous section is expressed through some function with parameters θ as $P_\theta(s_{t+1}|s_t, a_t)$. We can best imagine θ as a function that maps a state-action pair (s_t, a_t) to some output that then determines the distribution over s_{t+1}. Hence, we could have similarly expressed this function in function space view as $f(\cdot)$, where the dot $\cdot$ refers to the concatenated state-action domain, and the state-transition distribution as $P_{f(\cdot)}(s_{t+1}|s_t, a_t)$.

However, such function space transitions models are more difficult to sample from compared to ordinary weight space models. The reason is that state-transition models are auto-regressive, i.e. their output serves as input for the next time step, and it is not possible to sample a function from a process a priori without knowing where to evaluate that function (as opposed to weight space models). We refer the interested reader to Wilson et al. [2020] which could provide a remedy.

Under the former assumption, we need to determine a prior distribution over θ as $p_\gamma(\theta)$ with generative parameters γ. The likelihood is then simply defined as $p(s_1, ..., s_T|s_0, a_0, ...a_{T-1}, \theta) = \prod_{t=1}^{T} P_\theta(s_{t+1}|s_t, a_t)$. All that is left is the specification of an approximate posterior $q_\psi(\theta)$ with

variational parameters ψ in order to phrase the subsequent ELBO:

$$\text{ELBO}(\gamma, \psi) = \int q_\psi(\theta) \ln p(s_1, ..., s_T | s_0, a_0, ...a_{T-1}, \theta) \, \mathrm{d}\theta - \text{KL}\Big(q_\psi(\theta) \Big| \Big| p_\gamma(\theta)\Big) \quad (91)$$

$$= \sum_{t=0}^{T-1} \int q_\psi(\theta) \ln P_\theta(s_{t+1} | s_t, a_t) \, \mathrm{d}\theta - \text{KL}\Big(q_\psi(\theta) \Big| \Big| p_\gamma(\theta)\Big). \quad (92)$$

Note that we could have straightforwardly adopted a function space view by modeling the function prior $p_\gamma(f(\cdot))$ and approximate posterior $q_\psi(f(\cdot))$ with distributions over functions. This would require to swap $P_\theta(s_{t+1}|s_t, a_t)$ with $P_{f(\cdot)}(s_{t+1}|s_t, a_t)$ in Equations (91) and (92). Also note that due to the i.i.d. nature of the likelihood, it is not required to keep track of the temporal order of state-action-next-state tuples (s_t, a_t, s_{t+1}). It is in fact only required that each individual state-action-next-state tuple is temporally consistent, but the entirety of tuples is not required to represent a temporally consistent state-action trajectory. And since the entire procedure is action-conditioned, one also does not need to worry about how agent-environment interactions for training have been obtained, i.e. whether from a single agent or multiple agents with different policies does not matter.

Predicting future states $s_1^\star, s_2^\star, ..., s_{T^\star}^\star$ is then accomplished in an initial-state and action-conditioned fashion, given a single initial state $s_0^\star$ and a sequence of actions $a_0^\star, a_1^\star, a_2^\star, ..., a_{T^\star-1}^\star$:

$$p(s_1^\star, ..., s_{T^\star}^\star | s_0^\star, a_0^\star, ..., a_{T^\star-1}^\star) = \int \prod_{t=0}^{T^\star-1} P_\theta(s_{t+1}^\star | s_t^\star, a_t^\star) q_\psi(\theta) \, \mathrm{d}\theta, \quad (93)$$

where, importantly, the outer expectation is w.r.t. θ. This is practically crucial since a future state sequence can be generated by sampling θ from the approximate posterior $q_\psi(\theta)$ first, and then "unrolling" the state-transition model with the sampled θ. More concretely, given $s_0^\star$ and $a_0^\star$, one can sample $s_1^\star$ from $P_\theta(s_1^\star|s_0^\star, a_0^\star)$. Next, one can sample $s_2^\star$ from $P_\theta(s_2^\star|s_1^\star, a_1^\star)$ that depends on the previous state sample $s_1^\star$, and so forth, until one arrives at the final state $s_{T^\star}^\star$.

We conclude this section with a brief overview over the current state of the art in model-based reinforcement learning with a particular focus on robotics simulation domains because they provide a challenging benchmark (e.g. containing high-dimensional environments with highly non-linear state-transition distributions). We need to emphasize that a vast majority of the mentioned algorithms does not use approximate inference, but non-inference-based objectives in order to identify an environment model. However, since the model-learning component is a modular component of all model-based algorithms, we believe it could be straightforwardly replaced with a proper inference method. While conceptually more principled than non-inference objectives, it remains to be empirically validated if inference-based environment models yield better agent performance.

Most model-based reinforcement learning algorithms can be assigned to one of two categories: decision-time planning and background planning algorithms. Decision-time planners solve a virtual planning problem at every time step in order to identify a reasonable action, while background planners essentially comprise the vast majority of all other model-based algorithms that are not decision-time planners. Contemporary decision-time planners identify a suitable action for a given real environment state by generating several virtual state-action trajectories with the internal environment model starting at the observed state. They then evaluate all of these trajectories regarding their cumulative reward. Subsequently, a trajectory is selected proportional to its cumulative reward, i.e. the higher the cumulative reward of a trajectory the higher the probability of being chosen. Finally, the first action of the chosen trajectory is executed in the real environment. To generate virtual trajectories, an internal virtual policy is required. This is where contemporary algorithms divide into two categories: methods with state-unconditioned virtual policies [Chua et al., 2018, Nagabandi et al., 2018] and such with state-conditioned ones [Piche et al., 2019, Wang and Ba, 2019].

While background planning algorithms comprise a large class of different algorithms, we focus here on such background planning algorithms that train model-free algorithms virtually inside of an environment model because of their practical convenience. This idea is inspired by the "DYNA" architecture presented first in the seminal work of Sutton [1990] for tabular grid world scenarios. In contemporary model-based reinforcement learning algorithms for robotics simulation, two categories of background planners can be found. One class of algorithms trains "on-policy" model-free algorithms virtually inside the environment model [Kurutach et al., 2018, Luo et al., 2019] while the other trains "off-policy" model-free algorithms virtually [Janner et al., 2019]. The terms "on-policy"

and "off-policy" are fairly standard in the reinforcement learning literature and an explanation can be found in Sutton and Barto [1998]. In short, the difference between on-policy and off-policy is that the former needs to re-sample state-action trajectories at a high frequency and cannot naively make use of past trajectory samples, while the latter uses a replay buffer to utilize past trajectory samples at the prize of a possibly higher cumulative reward estimation bias.

For a broader overview of model-based algorithms for robotics simulation, we refer the interested reader to Wang et al. [2019]. Noteworthy mentioning explicitly is the work of Deisenroth and Rasmussen [2011] and Kamthe and Deisenroth [2018] that use GPs and a proper inference objective for model learning. Also worthwhile mentioning is that there has been recent progress for model-based reinforcement learning in visual domains such as Atari games [Leibfried and Vrancx, 2018, Kaiser et al., 2020, Schrittwieser et al., 2020].

5.5 VI for Partially Observable Model-Based Reinforcement Learning

The previous section assumes that environment states $s_t \in \mathcal{S}$ can be fully observed by the agent. However, in real-world scenarios, states are usually not fully observable. Instead, the agent needs to rely on noisy observations $o_t \in \mathcal{O}$ from which states s_t, that are in fact latent variables, need to be inferred. Often, o_t does not carry enough information to infer the latent state s_t precisely. This scenario is commonly referred to as "partially observable".

Mathematically, the partially observable setting follows the fully observable setting as described at the beginning of Section 5.3 but requires to extend the MDP definition by observations o_t. This is obtained via an observation distribution $P(o_t|s_t)$ that determines how observations o_t are generated from latent states s_t through a conditional probability distribution, yielding a partially observable MDP (POMDP) according to Kaelbling et al. [1998]. Note that the original POMDP definition specifies the observation distribution as $P(o_t|s_t, a_{t-1})$ with an additional dependence on the previous action a_{t-1} (from which we refrain in this tutorial in order to ease the exposition).
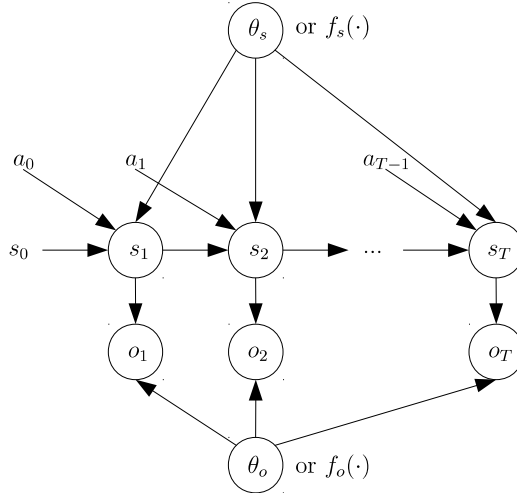

Figure 11: Graphical model for VI to infer the state-transition and observation distribution for model-based reinforcement learning in a partially observable setting. Environment states s_t and actions a_t are defined the same way as in the fully observable setting from Figure 10 with a time index $t \in \{0, 1, ..., T\}$. However, states s_t cannot be directly observed by the agent that needs to rely on noisy observations indicated by o_t. The unknown state-transition distribution is denoted as θ_s in parameter space view (or as $f_s(\cdot)$ in function space view), while the unknown observation distribution is denoted as θ_o or $f_o(\cdot)$ (in parameter space and function space view respectively). As in the fully observable case, inference is conditioned on the initial state s_0 as well as on all actions from a_0 up to a_{T-1}. Note that if the observation nodes o_t and the observation distribution node θ_o are removed, one would recover the graphical model from the fully observable case in Figure 10.

To phrase an inference problem, we need to extend the generative model from the fully observable setting as discussed in the previous section (and depicted in Figure 10) by observations o_t and

parameters θ_o for the observation distribution $P_{\theta_o}(o_t|s_t)$. Similar to earlier, θ_o can be imagined as a function that maps a latent state s_t to some output that determines the distribution over o_t. We could have alternatively expressed θ_o in function space view as $f_o(\cdot)$ and the observation distribution as $P_{f_o(\cdot)}(o_t|s_t)$. In order to distinguish parameters for the observation distribution $P_{\theta_o}(o_t|s_t)$ and state transition distribution $P_{\theta_s}(s_{t+1}|s_t, a_t)$, we have introduced the subscript s for state transition distribution parameters θ_s (and for the function space view representation $f_s(\cdot)$ respectively). The corresponding graphical model for the partially observable setting is depicted in Figure 11 and recovers the graphical model for the fully observable setting from Figure 10 when ignoring the nodes for the observations o_t and the observation distribution parameter θ_o.

Since θ_s, θ_o and $s_1, ..., s_T$ are all latent variables, we specify the prior of the generative model as $p_\gamma(\theta_s)p_\gamma(\theta_o)p(s_1, ..., s_T|s_0, a_0, ..., a_{T-1}, \theta_s)$ where γ refers to the entirety of all generative parameters and where we assume a factorized form as in earlier parts of this tutorial. The state prior is then defined as $p(s_1, ..., s_T|s_0, a_0, ..., a_{T-1}, \theta_s) = \prod_{t=1}^{T} P_{\theta_s}(s_{t+1}|s_t, a_t)$, and the likelihood as $p(o_1, ..., o_T|s_1, ..., s_T, \theta_o) = \prod_{t=1}^{T} P_{\theta_o}(o_t|s_t)$. Note that the likelihood is only conditioned on latent states s_1 up to s_T and the observation distribution parameters θ_o, but not on s_0 and the actions $a_0, ..., a_{T-1}$ as well as θ_s, due to the conditional independence assumptions in the graphical model. Under an approximate posterior of the form $q_\psi(\theta_s)q_\psi(\theta_o)q_\psi(s_1, ..., s_T)$ where $q_\psi(s_1, ..., s_T) = \prod_{t=1}^{T} q_\psi(s_t)$, we finally arrive at the following ELBO:

$$
\begin{aligned}
\text{ELBO}(\gamma, \psi) = & \int \int ... \int q_\psi(\theta_o)q_\psi(s_1, ..., s_T) \ln p(o_1, ..., o_T|s_1, ..., s_T, \theta_o) \, \mathrm{d}s_T... \, \mathrm{d}s_1 \, \mathrm{d}\theta_o \\
& - \text{KL}\Big(q_\psi(\theta_s)q_\psi(s_1, ..., s_T)\Big|\Big|p_\gamma(\theta_s)p(s_1, ..., s_T|s_0, a_0, ..., a_{T-1}, \theta_s)\Big) \\
& - \text{KL}\Big(q_\psi(\theta_o)\Big|\Big|p_\gamma(\theta_o)\Big) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (94) \\
= & \sum_{t=1}^{T} \int \int q_\psi(\theta_o)q_\psi(s_t) \ln P_{\theta_o}(o_t|s_t) \, \mathrm{d}s_t \, \mathrm{d}\theta_o \\
& - \sum_{t=1}^{T} \int \int q_\psi(\theta_s)q_\psi(s_{t-1})\text{KL}\Big(q_\psi(s_t))\Big|\Big|P_{\theta_s}(s_t|s_{t-1}, a_{t-1})\Big) \, \mathrm{d}s_{t-1} \, \mathrm{d}\theta_s \\
& - \text{KL}\Big(q_\psi(\theta_s)\Big|\Big|p_\gamma(\theta_s)\Big) - \text{KL}\Big(q_\psi(\theta_o)\Big|\Big|p_\gamma(\theta_o)\Big). \qquad\qquad (95)
\end{aligned}
$$

Note that the second line in Equation (94) expresses the KL of the joint distribution over θ_s and $s_1, ..., s_T$ between the approximate posterior and the prior (because the state transition distribution $P_{\theta_s}(s_{t+1}|s_t, a_t)$ depends on θ_s). This term then decomposes into two terms in Equation (95) (in line two and three). On a more subtle note, the second line of Equation (95) implicitly specifies $q_\psi(s_0)$ for notational convenience (because t starts at 1), which is defined as a Dirac delta function over the given initial state s_0 (and is hence not part of the variational model).

Predicting future observations $o_1^\star, o_2^\star, ..., o_{T^\star}^\star$ for a given initial state $s_0^\star$ and a given action sequence $a_0^\star, a_1^\star, a_2^\star, ..., a_{T^\star-1}^\star$ is then accomplished via the following:

$$
p(o_1^\star, ..., o_{T^\star}^\star|s_0^\star, a_0^\star, ..., a_{T^\star-1}^\star) = \int \int \int ... \int \prod_{t=0}^{T^\star-1} P_{\theta_o}(o_{t+1}^\star|s_{t+1}^\star)P_{\theta_s}(s_{t+1}^\star|s_t^\star, a_t^\star) \, \mathrm{d}s_{T^\star}... \, \mathrm{d}s_1
$$
$$
q_\psi(\theta_o)q_\psi(\theta_s) \, \mathrm{d}\theta_o \, \mathrm{d}\theta_s. \qquad\qquad (96)
$$

In order to sample an observation sequence, one first has to sample a state-transition model θ_s and an observation model θ_o from the approximate posteriors $q_\psi(\theta_s)$ and $q_\psi(\theta_o)$ respectively. Subsequently, one can "unroll" the trajectory starting from $s_0^\star$ and $a_0^\star$ by generating $s_1^\star \sim P_{\theta_s}(s_1^\star|s_0^\star, a_0^\star)$ and $o_1^\star \sim P_{\theta_o}(o_1^\star|s_1^\star)$, and repeating this procedure until one arrives at $s_{T^\star}^\star$ and $o_{T^\star}^\star$. Note that, in accordance with previous sections, the approximate state posterior $q_\psi(s_1, ..., s_T)$ is not used for prediction, but only used during training and "thrown away" afterwards.

In practice, it might be more important to infer latent states s_t on the fly at each time step t as the agent interacts with its environment, rather than generating new observation sequences. The reason why this might come in handy is that the agent could then use a state-conditioned policy $\pi(a_t|s_t)$ both for virtual planning but also for acting. One way of achieving this is to simply parameterize

the approximate posterior over latent states as $q_\psi(s_t|o_t)$ in an amortized fashion that maps observations o_t probabilistically to states s_t. While this has the advantage of enabling inference at act time, it ignores a lot of useful information at train time (since all the other observations and all actions carry information for inferring s_t). It is hence advisable to specify the approximate state posterior as $q_\psi(s_t|s_0, o_1, ..., o_T, a_0, ..., a_{T-1})$ with an explicit dependence on past and future time steps, and to learn a separate inference model for acting e.g. of the form $q_\chi(s_t|s_0, o_1, ..., o_t, a_0, ..., a_{t-1})$ that, importantly, does not require future time steps to infer the current latent state s_t at time step t. Here, χ indicates that the parameters for this inference model are not part of the ELBO objective and are hence not contained in the variational parameters ψ, but need to be identified through another objective (for example via filtering). How to precisely model variational posteriors in a POMDP setting is however outside the scope of this tutorial and we refer the reader to Tschiatschek et al. [2018] and Eleftheriadis et al. [2017] as examples for neural network and GP models respectively.

6 Summary

The aim of this tutorial is to provide access to and an overview over sparse GPs, VI and how to do VI with sparse GP models, targeting a broad audience of readers that are not familiar with neither GPs nor VI. The idea behind the outline of the manuscript is to introduce GPs and sparse GPs in Section 2 explicitly as stand-alone models only, and outside the scope of exact or approximate inference. Inference is then introduced in Section 3 and covers the main idea behind approximate inference with VI, however mostly focusing on weight space models for educational reasons. How to do VI with sparse GPs (that are function space models) is then finally covered in Section 4. Note that while Section 4 requires Sections 2 and 3 as necessary prerequisites, Sections 2 and 3 can themselves both be studied as stand-alone reading. The final Section 5 covers topics that we thought are important, but not as part of the core manuscript. Those topics present VI objectives in different scenarios ranging from unsupervised learning to reinforcement learning in the context of non-sequential and sequential decision-making (without and with environment models).

The main idea behind introducing sparse GPs in Section 2 is to see them as stand-alone models outside the scope of inference. We start with multivariate Gaussian distributions and conditioning operations in multivariate Gaussians (that also yield Gaussian distributions). This is necessary to understand sparse GPs which are essentially the consequence of conditioning GPs on a finite set of variables. It turns out that a proper exposition to the subject enables access to more recent advances in the field of GPs, such as interdomain GPs to provide more flexible features as well as computational gains, multioutput GPs that can handle multilabel problems, and deep GPs that hold the potential to model non-smooth functions (with which ordinary GPs might have problems).

Section 3 can be read as a stand-alone section and introduces the idea behind approximate inference using VI, with an emphasis on weight space models (such as neural networks for example) that we feel a large group of readers is familiar with (as opposed to function space models). We start with exact inference, its problems and how to provide a remedy with vanilla VI. But we subsequently also cover more advanced topics like importance-weighted VI to increase the quality of the inference routine (at the cost of an increased computational complexity), latent-variable VI to enable more expressive modelling, up to presenting a generic framework for Bayesian deep learning with Bayesian layers (that stack approximate inference blocks on top of each other).

VI with sparse GP models is eventually introduced in Section 4 and relies on previous sections as a prerequisite. We begin with how to do VI with ordinary sparse GPs and continue with extensions to latent-variable sparse GPs, importance-weighted sparse GPs, and deep sparse GPs. In the end, we culminate in a challenging multimodal synthetic example that demonstrates the flexibility of modeling when combining all the previously mentioned extensions, more precisely importance-weighting and latent variables in combination with deep GPs. Along the way, we provide several useful tips for experimenters for how to train sparse GP models in practice.

Further reaching topics in Section 5 conclude the manuscript with a few interesting VI objectives beyond vanilla supervised learning (which we have used as our default setting throughout the manuscript). Those examples comprise VI for unsupervised learning, but also for reinforcement learning showing how to recover non-sequential and sequential decision-making problems from an inference perspective. We also show how to learn environment models for sequential decision-making problems, both when the environment is fully and partially observable by the agent.

References

V Adam, S Eleftheriadis, A Artemev, N Durrande, and J Hensman. Doubly sparse variational Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2020.

M A Alvarez and N D Lawrence. Sparse convolved Gaussian processes for multi-output regression. In *Advances in Neural Information Processing Systems*, 2008.

M A Alvarez, D Luengo, and N D Lawrence. Latent force models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.

M A Alvarez, D Luengo, M Titsias, and N D Lawrence. Efficient multioutput Gaussian processes through variational inducing kernels. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010.

M A Alvarez, L Rosasco, and N D Lawrence. Kernels for vector-valued functions: a review. *Foundations and Trends in Machine Learning*, 2012.

M G Azar, V Gomez, and H J Kappen. Dynamic policy programming with function approximation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011.

C M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

K Blomqvist, S Kaski, and M Heinonen. Deep convolutional Gaussian processes. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2019.

C Blundell, J Cornebise, K Kavukcuoglu, and D Wierstra. Weight uncertainty in neural networks. In *Proceedings of the International Conference on Machine Learning*, 2015.

V Borovitskiy, A Terenin, P Mostowsky, and M P Deisenroth. Matern Gaussian processes on Riemannian manifolds. *arXiv*, 2020.

P Boyle and M Frean. Dependent Gaussian processes. In *Advances in Neural Information Processing Systems*, 2004.

T D Bui, J Yan, and R E Turner. A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation. *Journal of Machine Learning Research*, 2017.

Y Burda, R B Grosse, and R Salakhutdinov. Importance weighted autoencoders. In *Proceedings of the International Conference on Learning Representations*, 2016.

D R Burt, C E Rasmussen, and M van der Wilk. Rates of convergence for sparse variational Gaussian process regression. In *Proceedings of the International Conference on Machine Learning*, 2019.

D R Burt, C E Rasmussen, and M van der Wilk. Variational orthogonal features. *arXiv*, 2020.

K Chua, R Calandra, R McAllister, and S Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, 2018.

T M Cover and J A Thomas. *Elements of Information Theory*. Wiley & Sons, 2006.

A Damianou and N D Lawrence. Deep Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2013.

A Damianou, M Titsias, and N D Lawrence. Variational inference for latent variables and uncertain inputs in Gaussian processes. *Journal of Machine Learning Research*, 2016.

M P Deisenroth and C E Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the International Conference on Machine Learning*, 2011.

J Domke and D Sheldon. Importance weighting and variational inference. In *Advances in Neural Information Processing Systems*, 2018.

V Dutordoir, H Salimbeni, M P Deisenroth, and J Hensman. Gaussian process conditional density estimation. In *Advances in Neural Information Processing Systems*, 2018.

V Dutordoir, N Durrande, and J Hensman. Sparse Gaussian processes with spherical harmonic features. In *Proceedings of the International Conference on Machine Learning*, 2020a.

V Dutordoir, M van der Wilk, A Artemev, and J Hensman. Bayesian image classification with deep convolutional Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2020b.

D Duvenaud, O Rippel, R P Adams, and Z Ghahramani. Avoiding pathologies in very deep networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2014.

S Eleftheriadis, T F W Nicholson, M P Deisenroth, and J Hensman. Identification of Gaussian process state space models. In *Advances in Neural Information Processing Systems*, 2017.

R Fox, A Pakman, and N Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2016.

J R Gardner, G Pleiss, D Bindel, K Q Weinberger, and A G Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems*, 2018.

T Genewein, F Leibfried, J Grau-Moya, and D A Braun. Bounded rationality, abstraction, and hierarchical decision-making: An information-theoretic optimality principle. *Frontiers in Robotics and AI*, 2015.

J Grau-Moya, F Leibfried, T Genewein, and D A Braun. Planning with information-processing constraints and model uncertainty in Markov decision processes. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2016.

J Grau-Moya, F Leibfried, and H Bou-Ammar. Balancing two-player stochastic games with soft Q-learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2018.

J Grau-Moya, F Leibfried, and P Vrancx. Soft Q-learning with mutual-information regularization. In *Proceedings of the International Conference on Learning Representations*, 2019.

T Haarnoja, H Tang, P Abbeel, and S Levine. Reinforcement learning with deep energy-based policies. *Proceedings of the International Conference on Machine Learning*, 2017.

T Haarnoja, A Zhou, P Abbeel, and S Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine Learning*, 2018.

T Haarnoja, A Zhou, K Hartikainen, G Tucker, S Ha, J Tan, V Kumar, H Zhu, A Gupta, P Abbeel, and S Levine. Soft actor-critic algorithms and applications. *arXiv*, 2019.

J Hensman, N Fusi, and N D Lawrence. Gaussian processes for big data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2013.

J Hensman, A G G Matthews, M Filippone, and Z Ghahramani. MCMC for variationally sparse Gaussian processes. In *Advances in Neural Information Processing Systems*, 2015a.

J Hensman, A G G Matthews, and Z Ghahramani. Scalable variational Gaussian process classification. *Journal of Machine Learning Research*, 2015b.

J Hensman, N Durrande, and A Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 2018.

D Higdon. Space and space-time modeling using process convolutions. *Quantitative Methods for Current Environmental Issues*, 2002.

I Higgins, L Matthey, A Pal, C Burgess, X Glorot, M Botvinick, S Mohamed, and A Lerchner. Beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proceedings of the International Conference on Learning Representations*, 2017.

H Hihn and D A Braun. Specialization in hierarchical learning systems. *Neural Processing Letters*, 2020.

H Hihn, S Gottwald, and D A Braun. An information-theoretic on-line learning principle for specialization in hierarchical decision-making systems. In *Proceedings of the IEEE Conference on Decision and Control*, 2019.

M Janner, J Fu, M Zhang, and S Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, 2019.

A G Journel and C J Huijbregts. *Mining Geostatistics*. Academic Press, 1978.

L P Kaelbling, M L Littman, and A R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 1998.

L Kaiser, M Babaeizadeh, P Milos, B Osinski, R H Campbell, K Czechowski, D Erhan, C Finn, P Kozakowski, S Levine, A Mohiuddin, R Sepassi, G Tucker, and H Michalewski. Model-based reinforcement learning for Atari. In *Proceedings of the International Conference on Learning Representations*, 2020.

S Kamthe and M P Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2018.

M Karny. Fully probabilistic design unifies and supports dynamic decision making under uncertainty. *Information Sciences*, 2020.

D P Kingma and M Welling. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations*, 2014.

D P Kingma, D J Rezende, S Mohamed, and M Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, 2015.

T Kurutach, I Clavera, Y Duan, A Tamar, and P Abbeel. Model-ensemble trust-region policy optimization. In *Proceedings of the International Conference on Learning Representations*, 2018.

T Lattimore and C Szepesvari. *Bandit Algorithms*. Cambridge University Press, 2020.

M Lazaro-Gredilla and A Figueiras-Vidal. Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems*, 2009.

F Leibfried and D A Braun. A reward-maximizing spiking neuron as a bounded rational decision maker. *Neural Computation*, 2015.

F Leibfried and D A Braun. Bounded rational decision-making in feedforward neural networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2016.

F Leibfried and J Grau-Moya. Mutual-information regularization in Markov decision processes and actor-critic learning. In *Proceedings of the Conference on Robot Learning*, 2019.

F Leibfried and P Vrancx. Model-based regularization for deep reinforcement learning with transcoder networks. In *NIPS Workshop*, 2018.

F Leibfried, J Grau-Moya, and H Bou-Ammar. An information-theoretic optimality principle for deep reinforcement learning. In *NIPS Workshop*, 2018.

F Leibfried, S Pascual-Diaz, and J Grau-Moya. A unified Bellman optimality principle combining reward maximization and empowerment. In *Advances in Neural Information Processing Systems*, 2019.

S Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv*, 2018.

Y Luo, H Xu, Y Li, Y Tian, T Darrell, and T Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *Proceedings of the International Conference on Learning Representations*, 2019.

A G G Matthews, J Hensman, R Turner, and Z Ghahramani. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2016.

A G G Matthews, M van der Wilk, T Nickson, K Fujii, A Boukouvalas, P Leon-Villagra, Z Ghahramani, and J Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 2017.

C A Micchelli and M Pontil. On learning vector-valued functions. *Neural Computation*, 2005.

O Nachum, M Norouzi, K Xu, and D Schuurmans. Bridging the gap between value and policy based reinforcement learning. *Advances in Neural Information Processing Systems*, 2017.

A Nagabandi, G Kahn, R S Fearing, and S Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2018.

G Neu, V Gomez, and A Jonsson. A unified view of entropy-regularized Markov decision processes. *arXiv*, 2017.

A O'Hagan. Some Bayesian numerical analysis. *Bayesian Statistics*, 1992.

P A Ortega and D A Braun. Thermodynamics as a theory of decision-making with information-processing costs. *Proceedings of the Royal Society A*, 2013.

P A Ortega, D A Braun, J Dyer, K-E Kim, and N Tishby. Information-theoretic bounded rationality. *arXiv*, 2015.

Z Peng, T Genewein, F Leibfried, and D A Braun. An information-theoretic on-line update principle for perception-action coupling. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

J C Petangoda, S Pascual-Diaz, V Adam, P Vrancx, and J Grau-Moya. Disentangled skill embeddings for reinforcement learning. In *NIPS Workshop*, 2019.

J Peters, K Mulling, and Y Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010.

A Piche, V Thomas, C Ibrahim, Y Bengio, and C Pal. Probabilistic planning with sequential Monte Carlo methods. In *Proceedings of the International Conference on Learning Representations*, 2019.

C E Rasmussen and C K I Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

D J Rezende, S Mohamed, and D Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*, 2014.

G Riutort-Mayol, P-C Buerkner, M R Andersen, A Solin, and A Vehtari. Practical Hilbert space approximate Bayesian Gaussian processes for probabilistic programming. *Computing Research Repository*, 2020.

J Rubin, O Shamir, and N Tishby. Trading value and information in MDPs. In *Decision Making with Imperfect Decision Makers*. Springer, 2012.

H Salimbeni and M P Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, 2017.

H Salimbeni, V Dutordoir, J Hensman, and M P Deisenroth. Deep Gaussian processes with importance-weighted variational inference. In *Proceedings of the International Conference on Machine Learning*, 2019.

A D Saul, J Hensman, A Vehtari, and N D Lawrence. Chained Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2016.

J Schrittwieser, I Antonoglou, T Hubert, K Simonyan, L Sifre, S Schmitt, A Guez, E Lockhart, D Hassabis, T Graepel, T Lillicrap, and D Silver. Mastering Atari, Go, Chess and Shogi by planning with a learned model. *arXiv*, 2020.

J Schulman, S Levine, P Moritz, M Jordan, and P Abbeel. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning*, 2015.

J Schulman, P Abbeel, and X Chen. Equivalence between policy gradients and soft Q-learning. *arXiv*, 2017.

C A Sims. Rational inattention and monetary economics. In *Handbook of Monetary Economics*. Elsevier, 2010.

K Sohn, H Lee, and X Yan. Learning structured output representation using deep conditional generative models. *Advances in Neural Information Processing Systems*, 2015.

R S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the International Conference on Machine Learning*, 1990.

R S Sutton and A G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

S Tiomkin and N Tishby. A unified Bellman equation for causal information and value in Markov decision processes. In *arXiv*, 2018.

N Tishby and D Polani. Information theory of decisions and actions. In *Perception-Action Cycle*. Springer, 2011.

M Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.

D Tran, M W Dusenberry, M van der Wilk, and D Hafner. Bayesian layers: A module for neural network uncertainty. In *Advances in Neural Information Processing Systems*, 2019.

S Tschiatschek, K Arulkumaran, J Stuhmer, and K Hofmann. Variational inference for data-efficient model learning in POMDPs. *arXiv*, 2018.

M van der Wilk and C E Rasmussen. Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems*, 2017.

M van der Wilk, V Dutordoir, S T John, A Artemev, V Adam, and J Hensman. A framework for interdomain and multioutput Gaussian processes. *arXiv*, 2020.

T Wang and J Ba. Exploring model-based planning with policy networks. *arXiv*, 2019.

T Wang, X Bao, I Clavera, J Hoang, Y Wen, E Langlois, S Zhang, G Zhang, P Abbeel, and Y Ba. Benchmarking model-based reinforcement learning. *arXiv*, 2019.

F Wenzel, K Roth, B Veeling, J Swiatkowski, L Tran, S Mandt, J Snoek, T Salimans, R Jenatton, and S Nowozin. How good is the Bayes posterior in deep neural networks really? In *Proceedings of the International Conference on Machine Learning*, 2020.

J T Wilson, V Borovitskiy, A Terenin, P Mostowsky, and M P Deisenroth. Efficiently sampling functions from Gaussian process posteriors. In *Proceedings of the International Conference on Machine Learning*, 2020.

B D Ziebart. *Modeling purposeful adaptive behavior wih the principle of maximum causal entropy*. PhD thesis, Carnegie Mellon University, USA, 2010.