Markus Kaiser

Dissertation

Structured Models with Gaussian Processes

Chair for Foundations of Software Reliability
and Theoretical Computer Science
Department of Informatics
Technical University of Munich

Learning Systems Group
Technology
Siemens AG

Structured Models with Gaussian Processes

Markus Kaiser

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 11.01.2021 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 27.04.2021 angenommen.

Zusammenfassung

Mit Methoden des maschinellen Lernens konnten in den letzten Jahren in einer Vielzahl von digitalen Anwendungsbereichen wie Spracherkennung, Computer-Vision oder Videospielen beeindruckende Erfolge erzielt werden. Der Transfer zu Anwendungen in der physikalischen Welt hat sich jedoch als eine Herausforderung erwiesen, da sie eine Reihe neuer Anforderungen mit sich bringen. Lernverfahren müssen Expertenwissen effizient nutzen, mit wenigen Daten auskommen und Unsicherheiten bestimmen. In dieser Dissertation wird untersucht, wie strukturierte probabilistische Modelle es uns ermöglichen, diesen Anforderungen gerecht zu werden. Strukturierte Modelle kombinieren datengestützte und theoriegetriebene Modellierungsansätze, um Expertenwissen zu formalisieren und dennoch neue Erkenntnisse aus Daten zu gewinnen.

In dieser Arbeit formulieren wir probabilistische strukturierte Modelle mithilfe von bayesschen nicht-parametrischen Methoden. Wir nutzen Informationen über die Struktur des Lernproblems, um Modelle zu formulieren, die Wissen reproduzieren, für Domänenexperten verständlich sind, physikalisch plausible Vorhersagen in unbekannten Situationen liefern und ihre eigene Unsicherheit beziffern können. Dazu betten wir allgemeine Funktionsapproximatoren in probabilistische Modelle ein und formulieren Inferenzmethoden auf Basis von verketteten Gaußprozessen. Am Beispiel realer industrieller Anwendungen wie der Erkennung fehlerhafter Sensoren und der Vorhersage der Stromerzeugung eines Windparks zeigen wir, dass strukturierte Modelle informative Unsicherheiten liefern, interpretierbar sind und erfolgreich generalisieren.

In Situationen, in denen die interne Struktur und das Generalisierungsverhalten von Modellen in den Mittelpunkt rücken, kann Modellselektion basierend auf klassischen Metriken unzureichend sein, um bevorzugte Modelle zu identifizieren. Wir formalisieren den subjektiven Anteil der Modellselektion, indem wir die Aufgabe, die ein Modell lösen soll, in die Auswahl miteinbeziehen. Wir zeigen an einem Reinforcement-Learning Problem, dass semantisch korrekte Modelle andere Modelle mit ähnlichen Metriken übertreffen und es Experten ermöglichen, das Verhalten von Agenten zu beeinflussen. Wir untersuchen die Eigenschaften strukturierter Modelle in einem breiteren Kontext, die Grenzen gängiger Inferenzverfahren und disktieren, warum Modelle mit suboptimalen Metriken in hierarchischen Systemen erfolgreich eingesetzt werden können und wie bayessche Inferenzprobleme formulierten werden können, die nachgelagerte Aufgaben berücksichtigen.

Abstract

Machine learning methods have seen great success recently in a wide range of digital domains such as speech recognition, computer vision or video games. However, bridging the gap to applications in the physical world has proved to be challenging as they introduce a new set of requirements. Machine learning systems must make efficient use of expert knowledge, handle low data regimes, and quantify uncertainties. This thesis explores how structured probabilistic models allow us to cope with these requirements. Structured models combine black-box and white-box modeling approaches to formalize expert knowledge while still being able to gain new insights from data.

In this work, we formulate Bayesian structured models using methods from Bayesian nonparametrics. We use information about the structure of a learning problem to formulate machine learning models that reproduce knowledge, are understandable for domain-experts, make physically plausible predictions in unseen situations and can quantify their own uncertainty. We explore how to embed general function approximators in Bayesian probabilistic models to enforce structure and discuss how to formulate inference schemes based on composite and hierarchical Gaussian process models. Using real-world industrial applications such as the detection of faulty sensors and the prediction of power generation in a wind-farm as examples, we show how to use structured models to factorize uncertainties, achieve interpretability, and generalize to unobserved inputs.

In settings where internal structure and generalization behavior come into focus, model selection using marginal likelihoods can be insufficient to identify desirable models. We consider how to formalize the subjectiveness in model selection through the task a model will be used to solve. We show that in a reinforcement learning problem, semantic models outperform other models with similar performance metrics and allow experts to influence agent behavior. We explore the properties of structured models in a broader context and discuss the limits of current inference schemes, why models with suboptimal marginal likelihoods can perform well in hierarchical systems, and how to formulate Bayesian inference problems that take downstream tasks into account.

Acknowledgements

I want to thank my supervisor Prof. Dr. Thomas A. Runkler for his guidance and support. Thomas has always encouraged me to explore my ideas while offering invaluable advice and making sure I do not lose focus. I am very grateful to my co-supervisor Prof. Dr. Carl Henrik Ek for his enthusiasm and mentorship. He is an inspiring teacher and shows me what is important in academia. Thank you for welcoming me to your research group, making Bristol a second home for me, and the countless discussions about the big picture and the small details. I also want to thank my advisor Dr. Clemens Otte for his valuable input and his exceptional ability to combine research and applications. I could not have asked for better supervision or a more encouraging research environment during my studies.

I am thankful to everyone at Siemens and the Learning Systems group, especially Volkmar Sterzing for giving us PhD candidates the freedom to pursue our interests and enabling us to work on exciting industrial applications. I am grateful to Steffen Udluft and Hans Georg Zimmermann for always taking the time to discuss new ideas and for broadening my horizon as well as my fellow PhD candidates Stefan Depeweg, Daniel Hein and Phillip Swazinna for many discussions, whiteboard-drawings and meetings at the Biergarten. Thanks to Marion Eigner for her refreshing perspective and friendship.

I feel lucky to be part of a motivating and unique group in Bristol and Bath. Thank you to Prof. Dr. Neill Campbell for your passion for good research and for your help in pursuing worthwhile ideas. I want to thank Erik Bodin, Ieva Kazlauskaite and Ivan Ustyuzhaniov for many thoughtful discussions and collaborations.

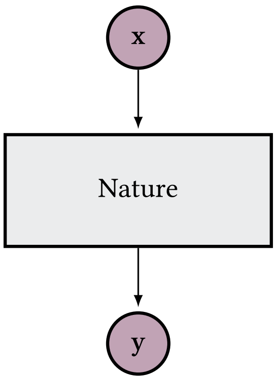
A big thank you to my siblings Matthias and Andrea for your support and encouragement. I sincerely thank my parents, Pia and Robert, for enabling and inspiring us to follow our dreams and for your unconditional support. Above all I want to thank Hannah for her patience and understanding while being in COVID-19 lockdown with somebody writing their thesis. Thank you for all the lunches, the much-needed distractions, the happiness, and your love.

Contents

Chapter 1

Introduction

In many machine learning problems, the task is to derive a model for observational data that has been generated by some process in nature that is not fully understood. Starting with an input-vector $\mathbf{x}$ describing its initial state, nature produces an output-vector $\mathbf{y}$. Given a set of observational pairs $(\mathbf{x}, \mathbf{y})$, we want to learn about the underlying, possibly stochastic, functional dependency

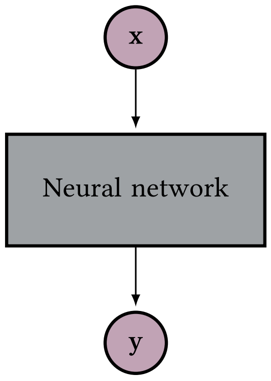


$$\mathbf{y} = f(\mathbf{x}). \tag{1.1}$$

The goal of learning is often to either gain new knowledge about nature by describing the data or to be able to predict outputs $\mathbf{y}_*$ for previously unseen inputs $\mathbf{x}_*$. While traditional statistics favor the white-box modeling culture that focuses on explaining observational data using causal theoretic models, machine learning and specifically deep learning is more closely associated with the black-box culture, which focuses on prediction instead [19, 99]. The impressive successes of both cultures have been achieved in different application domains, however, which is due to their respective pros and cons.

1.1 Black-Box Models in Deep Learning

When formulating a black-box model, we accept that we cannot model nature directly and focus on the functional dependency f instead. The approach is to find an algorithm $f_\mathbf{w}$ with parameters $\mathbf{w}$ that approximates the true dependency f in



$$\mathbf{y} = f_\mathbf{w}(\mathbf{x}). \tag{1.2}$$

Using general function approximators like neural networks with a typically large set of weights $\mathbf{w}$, any smooth dependency can be modeled given enough observations. With growing computational power and the availability of data sets, black-box approaches have led to impressive advancements in domains that are hard to formalize. Successes have been seen in a wide range of digital tasks such as speech recognition [7, 24, 60], computer vision [76, 92], non-cooperative games [11, 100], or machine translation [63]. The abundance of data in these domains shifts the focus away from traditional statistical models to highly adaptable and scalable approaches. Black-box models do not claim to uncover or represent knowledge about the true process in nature but capture correlations between inputs and outputs instead. Their advantages and disadvantages can be summarized as follows.

Pros (Deep learning)

Universal approximation Neural networks can potentially find useful structure in any data set, even if the problem domain is not well-understood or cannot be formalized by experts.

Simple portability Black-box model formulations can be applied to many different applications due to their assumed independence of the concrete problem.

Strong scalability The internal structure of black-box models can often be chosen to leverage computational power optimally. Models in deep learning can cope with billions of parameters and observations.

Cons (Deep learning)

Data bias Black-box approaches can only uncover the structure that is present in the training data. They, therefore, typically require a large amount of training data in all parts of the system.

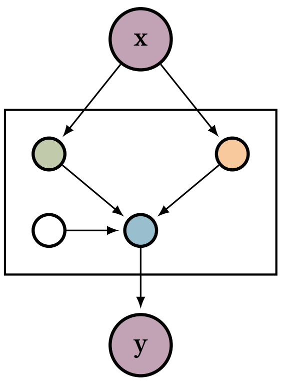
Weak interpretability Black-box models are not designed to offer causal explanations of observations. As a result, the complex internal structure of models in deep learning is generally not understandable to human experts [91].

Unclear generalization A black-box model can only be evaluated in terms of its ability to predict unseen data. In parts of a system where no data is available, it is hard to make statements about when or how a model generalizes.

The highly adaptable and scalable models from deep learning excel at finding structure in large and complex data sets. However, the black-box approach struggles in environments where the model's internal structure or the generalization behavior come into focus.

1.2 White-Box Models in Bayesian Statistics

White-box models are based on strong and human-interpretable internal structure. The focus is on describing and understanding observational data and thus the true generative process in nature. In modern science, white-box models are used in large-scale experiments such as in particle physics [107] or astronomy [27, 28]. In such experiments, noisy and limited data play a major role. To cope with imperfect knowledge, causal theoretic models are extended with statistical theory to formulate probabilistic models. Probabilistic models consider the likelihood $p(\mathbf{y}\,|\,\mathbf{x})$ based on knowledge about the model components 🟢, 🟠, … in



$$p(\mathbf{y}\,|\,\mathbf{x}) = \int p(\mathbf{y}, \text{🟢}, \text{🟠}, \text{🔵}, \text{⚪}\,|\,\mathbf{x})\, d\text{🟢}\, d\text{🟠}\, d\text{🔵}\, d\text{⚪}. \tag{1.3}$$

Bayesian inference offers a framework for formalizing this knowledge in prior assumptions and combining them with data. Using Bayes' rule, a distribution of plausible models is reweighed by their ability to generate data to yield a posterior distribution p(●, ●, ●, ○ | $\mathbf{x}, \mathbf{y}$) over plausible explanations. To make predictions about novel observations $p(\mathbf{y}_* \mid \mathbf{x}_*)$, all plausible models are taken into account, allowing us to both quantify confidence in predictions and take more unlikely but relevant scenarios into consideration. Scientific white-box models are driven by strong theory that provides causal explanations. Observational data is used to answer specific questions about a small number of hypotheses, leading to the following pros and cons.

Pros (Bayesian statistics)

Strong interpretability Since a white-box model is constructed from human-interpretable components and causal theory, model behavior and predictions are understandable to experts.

Trustworthy predictions Bayesian probabilistic models quantify their confidence about predictions in a principled manner based on explicit assumptions.

Safe generalization The generalization of white-box models is driven by human-interpretable causal theory that can be evaluated and verified by experts.

Cons (Bayesian statistics)

Model bias All insights from white-box models are with respect to strong causal theoretic modeling assumptions. Wrong or insufficient assumptions cannot be compensated through data, limiting the applicability of white-box models in problem domains that are hard to formalize.

Subjective model-selection White-box models aim to model causal relationships in nature. It is a hard problem to select the correct model from a set of plausible explanations [112].

Weak scalability The theory behind white-box models is often problem-specific and hard to transfer to other applications. Similarly, complex theory often requires specialized learning algorithms that do not scale to large amounts of data.

White-box models have been the driving force behind many advances in economics, medicine, and the natural sciences and allowed researchers to find and understand the structure in complex and heterogenous data [44, 50]. However, as processes get more complex, and the amount of available data grows, formulating white-box models becomes a very difficult task [110].

1.3 Structured Models with Gaussian Processes

White-box and black-box modeling approaches serve different purposes in data analysis. White-box models are used in applications where hypotheses should be tested when a strong theoretical background is available. Black-box models are most successful when correlations and associations in large amounts of data should be used to make predictions about systems where the consequences of mistakes are mild. However, for many safety-critical applications of machine learning in the physical world, the situation is different. Complete theoretic models are not available to solve a learning task, and new insights need to be inferred from potentially large amounts of data. Simultaneously, models need to make physically plausible predictions and are required to be interpretable for domain experts.

Take wind power production as an example: To optimize the efficiency of a wind-turbine, the speed and pitch have to be controlled according to the local wind conditions. In a wind-farm, turbines are typically equipped with sensors for wind speed and direction. The goal is to use these sensor data to produce accurate estimates and forecasts of the wind conditions at every turbine in the farm. For the ideal case of a homogeneous and very slowly changing wind field, wind conditions can be estimated using the propagation times computed from geometry, wind speed and direction. In the real world, however, wind fields are not homogeneous, exhibit global and local turbulence, and interfere with the turbines and the terrain inside and outside the farm. This makes it extremely difficult to construct accurate white-box models of wind propagation in a farm. At the same time, data-driven black-box approaches struggle due to identifiability issues introduced by the noise of the unpredictable turbulence. While we cannot formulate analytical models, domain experts do have an intuitive understanding of the underlying physics of wind propagation. A successful model needs to combine the white-box and black-box approaches: It needs to reproduce expectations like slowly changing prevailing wind conditions or gusts that travel through the system while still being able to infer new knowledge about specific turbine behavior or the influence of terrain from data. By checking if these expectations are met, domain experts can build trust in the model's predictions for novel situations.

In this work, we study how to formulate structured Bayesian models that reproduce knowledge, are understandable for domain experts, and can still gain new insights from data. Table 1.1 shows that structured models need to combine the advantages of the general function approximators from deep learning and the strong and interpretable structure from Bayesian statistics to fulfill all requirements of safety-critical systems. There are many approaches in machine learning that combine some but not all of these

Table 1.1: The model properties considered as research questions in this thesis.

	Deep learning	Bayesian statistics
Data-driven insights	✓	✗ RQ1
Strong scalability	✓	✗
Interpretable results	RQ2 ✗	✓
Trustworthy predictions	✗	✓
Semantic model-selection	✗	✗ RQ3

properties. Rule-based or equation-based systems [54, 71, 82] can combine data-driven insights with interpretable results, but they struggle with representing uncertainties and making trustworthy predictions in poorly understood systems. Extensions to deep learning such as Bayesian neural networks (BNNs) [38, 47] augment neural networks with priors over weights to yield predictive uncertainties. However, BNNs remain black-box models with unclear generalization behavior and low interpretability [91]. In this thesis, we embed Gaussian processes (GPs) in hierarchical probabilistic models. GPs are well-understood non-parametric distributions over functions that allow us to encode Bayesian assumptions explicitly. The combination of constraints imposed by hierarchical Bayesian structure and the data-driven insights provided by the embedded general approximators enables principled and interpretable reasoning about data-driven insights. The research questions we consider in this thesis are as follows.

RQ1: How can we reduce the model-bias of white-box models and efficiently learn from data? Starting from a fully white-box model, we explore how to relax a strong theoretical structure by embedding general function approximators in a hierarchy. While conserving interpretability and informative uncertainty quantification, we use GPs to learn about problem-components we cannot formalize. We discuss how to formulate inference schemes that scale to large data sets.

RQ2: How can we reduce the data-bias of black-box models and reliably reproduce expert expectations? Starting from black-box deep GP models, we study how to add Bayesian structure to reproduce expert knowledge and factorize uncertainties. The additional structure allows us to formulate problem-driven constraints for desirable solutions that yield physically plausible results. We explore how rich internal structure helps models to generalize to novel situations in an interpretable manner.

RQ3: How can we formalize model selection in situations where internal model structure and generalization behavior come into focus? As evaluation data in critical parts of the system is often scarce in physical environments, performance measures such as the generalization error are not enough to identify desirable structured models. Simultaneously, the subjective model-selection of white-box approaches is problematic in domains that are not well-understood. To formalize the description of desirable structured models, we explore how downstream tasks can be taken into account for model selection.

1.4 Contributions

This work presents novel Bayesian structured models inspired by industrial applications. The main contributions to the research questions in Table 1.1 are as follows. They appeared in a number of peer-reviewed publications and patents we outline below.

RQ1 In Contributions 2, 9 and 10, we formulate a Bayesian interpretation of the data association problem. We discuss an application of this model to the prediction of the combustion dynamics of a gas turbine, where the structured model allows us to distinguish between different dynamical regimes.

RQ2 In Contributions 1, 7 and 8, we formulate a hierarchical Bayesian model for a nonlinear time-series alignment problem. We show that this model is capable of representing the complex interactions between turbines in a wind farm, and discuss how it can be used to derive more efficient controllers for wind-turbines.

RQ3 In Contributions 3 and 4, we apply our data association model to a reinforcement learning task and show how a semantic decomposition of the dynamics reduces the data requirements, and produces interpretable solutions. In Contribution 5, we formulate surrogate models for Bayesian optimization problems that focus on the informative structure of the objective functions by sacrificing local accuracy and in Contribution 6, we present an argument for why inference schemes based on factorizations between layers cannot represent heterogeneous posteriors.

Own Publications

1. Markus Kaiser, Clemens Otte, Thomas A. Runkler, and Carl Henrik Ek. “Bayesian Alignments of Warped Multi-Output Gaussian Processes”. In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle,

K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 6995–7004. arXiv: 1710.02766

2. Markus Kaiser, Clemens Otte, Thomas A. Runkler, and Carl Henrik Ek. "Data Association with Gaussian Processes". In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD) 2019*. Sept. 2019. arXiv: 1810.07158

3. Markus Kaiser, Clemens Otte, Thomas A. Runkler, and Carl Henrik Ek. "Interpretable Dynamics Models for Data-Efficient Reinforcement Learning". In: *Computational Intelligence and Machine Learning* ESANN 2019 proceedings (2019), p. 6

4. Markus Kaiser, Clemens Otte, Thomas A. Runkler, and Carl Henrik Ek. "Bayesian Decomposition of Multi-Modal Dynamical Systems for Reinforcement Learning". In: *Neurocomputing* (Apr. 10, 2020). ISSN: 0925-2312. DOI: 10.1016/j.neucom.2019.12.132

5. Erik Bodin, Markus Kaiser, Ieva Kazlauskaite, Zhenwen Dai, Neill D. F. Campbell, and Carl Henrik Ek. "Modulating Surrogates for Bayesian Optimization". In: *Proceedings of the International Conference on Machine Learning (ICML) 119*. Feb. 24, 2020. arXiv: 1906.11152

6. Ivan Ustyuzhaninov, Ieva Kazlauskaite, Markus Kaiser, Erik Bodin, Neill D. F. Campbell, and Carl Henrik Ek. "Compositional Uncertainty in Deep Gaussian Processes". In: *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*. Feb. 25, 2020. arXiv: 1909.07698

Patents

7. Per Egedal, Peder Bay Enevoldsen, Alexander Hentschel, Markus Kaiser, Clemens Otte, Volkmar Sterzing, Steffen Udluft, and Marc Christian Weber. "Verfahren und Vorrichtung zur kooperativen Steuerung von Windturbinen eines Windparks". European pat. 3517774A1. Siemens Gamesa Renewable Energy As. July 31, 2019

8. Markus Kaiser and Marc Christian Weber. "Verfahren und Vorrichtungen zur automatischen Ermittlung und/oder Kompensation des Einflusses einer Wirbelschleppe auf eine Windkraftanlage". European pat. 3527817A1. Siemens AG. Aug. 21, 2019

9. Markus Michael Geipel, Thomas Hubauer, Markus Kaiser, and Anja von Beuningen. “Transferlernen von Modellen des maschinellen Lernens unter Verwendung einer Wissensgraphdatenbank”. European pat. 3620997A1. Siemens AG. Mar. 11, 2020

10. Stefan Depeweg, Markus Michael Geipel, Markus Kaiser, and Steffen Udluft. “Computerimplementiertes Verfahren zum Abschätzen eines technischen Verhaltens einer Vorrichtung”. European pat. 3623881A1. Siemens AG. Mar. 18, 2020

1.5 Thesis Outline

The content of this thesis is structured as follows. In Chapter 2, we provide a short introduction to Bayesian machine learning and Bayesian nonparametrics with a focus on Gaussian processes (GPs). We discuss the connection between statistical learning theory and Bayesian inference and their application to hierarchical models. After introducing GPs, we discuss variational approaches to sparse GPs, their extension to hierarchical GP models, and efficient inference schemes.

In Chapter 3, we consider the first research question and explore how to formulate structured hierarchical models using the data association problem as an example. We separate the data association problem into hierarchical components and derive an efficient joint inference scheme, resulting in a fully Bayesian model. We show that this model can factorize multi-modal data into independent processes, providing explicit models for both the processes and assignment probabilities. Comparisons with previous models show the additional qualitative model capabilities of the structured approach and competitive black-box performance. However, experiments also show that standard measures are not enough to identify desirable models.

In Chapter 4, we consider the second research question and formulate a structured Bayesian model that reproduces expert knowledge about wind propagation in a windfarm. We interpret the problem of modeling the power production of multiple windturbines as a nonlinear alignment problem and derive an efficient inference scheme. Based on expert knowledge about the underlying latent and turbulent wind field, we can derive a structured model that enforces a physically plausible dependency structure between multiple time-series. We discuss how these constraints help solve an otherwise highly ambiguous learning problem and show that the imposed structure leads to a rich internal model-structure that experts can interpret.

In Chapter 5, we consider the third research question and discuss how to formalize the subjectiveness introduced via expert knowledge and model ambiguities to evaluate structured models. We include the task a model will be used to solve into model selection, which allows us to distinguish between qualitatively different models showing similar performance metrics. We revisit the data association problem and embed it into a reinforcement learning problem, where identifying the correct underlying dynamics, and therefore a desirable model, is critical to finding a successful policy. We show that semantic hierarchical structure increases data efficiency and allows domain experts to influence agent behavior through detailed insights into the dynamics of a system.

Chapter 6 concludes the work presented in this thesis and interprets the results in a broader context by further exploring the properties and evaluation of structured hierarchical models. In Section 6.1, we present an intuitive argument for why inference schemes based on factorizations between layers cannot represent heterogeneous posteriors. In Section 6.2, we argue why models with suboptimal marginal likelihoods can perform well in hierarchical systems. In Section 6.3, we explore this idea further and consider how tasks can be included in the inference problem directly. Finally, we discuss possible further directions for research.

Chapter 2

Preliminaries

In Chapter 1, we described the black-box and white-box modeling cultures and introduced the idea of structured models. Structured models share rich internal structure with white-box models while accepting that some aspects of a learning problem cannot be modeled in full, thereby introducing black-box model components. In this thesis, we will use ideas from Bayesian nonparametrics to formulate structured models. In this chapter, we introduce the theoretical foundations of Bayesian machine learning and then introduce Gaussian process models, a tool for Bayesian nonparametric function approximation.

We first provide a short introduction to statistical learning theory and Bayesian machine learning. We then introduce Gaussian processes formally, describe how to formulate and select a GP prior, and how to derive a GP posterior. Since GPs defined on many observations are computationally expensive, sparse approximations are used in practice. We discuss one class of sparse approximations based on inducing observations and introduce approaches based on variational inference. Next, two extensions to variational inference in hierarchical GP models are introduced that will be used to formulate more complex hierarchical models containing multiple Gaussian processes in the next chapters.

2.1 Machine Learning Problems

One of the roots of machine learning (ML) lies in the study of algorithms in theoretical computer science. An algorithm is a well-defined sequence of computational steps transforming a set of inputs to a set of outputs. It is a tool for solving a computational problem, which is defined by an abstract problem of admissible inputs and expected outputs. An algorithm solves such a problem if, for every possible input, the algorithm provably yields the correct output.

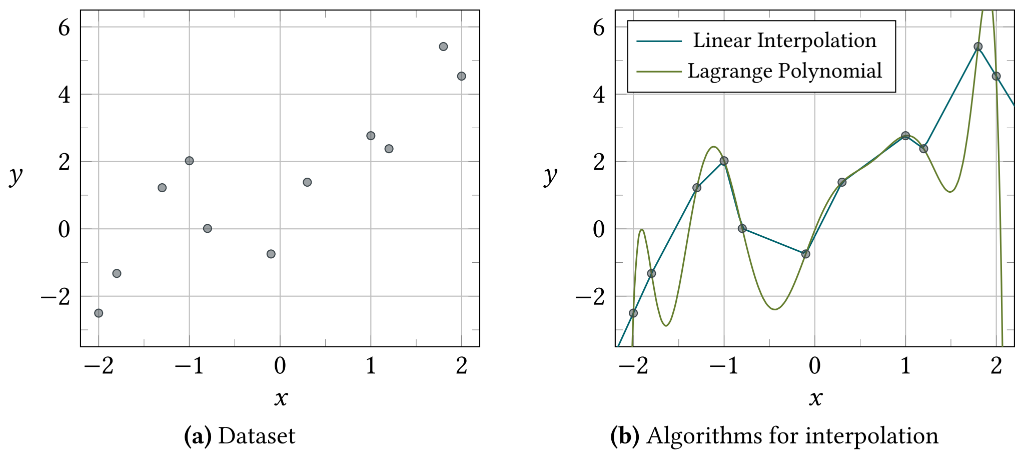


Figure 2.1: Computational problems defined on a dataset have a well-defined and unique solution. This solution can be characterized through algorithms such as linear interpolation or Lagrange polynomials.

Consider the computational problem of sorting a list of numbers in ascending order. One possible formulation [29] of the sorting problem is:

Problem 1 (Sorting)
Input: A sequence of N integers $\mathbf{I} = (i_1, \dots, i_N)$

Output: A reordering of $\mathbf{I}$ called $\mathbf{O} = (o_1, \dots, o_N)$ such that $o_1 \leq \dots \leq o_N$.

For example, for the input $\hat{\mathbf{I}} = (12, 8, 23, 4)$ the correct output is $\hat{\mathbf{O}} = (4, 8, 12, 23)$. A concrete input $\hat{\mathbf{I}}$ is called an instance of a problem. Importantly, such an instance contains all the required information to compute the unique output $\hat{\mathbf{O}}$. The correctness of the output can be checked via the formal problem. The various available (correct) sorting algorithms only differ in which and how many computational steps they take to arrive at the output, not in the output itself.

Machine learning can be seen as an extension of algorithmics towards problems where a formal description of a uniquely defined solution does not exist. Instead, problems in ML are characterized by the observation of finitely many examples or data together with the objective to derive knowledge about how these examples were generated. This knowledge can then be used to describe common patterns in the data or generate new examples that conform to previous observations.

As an example, consider the data shown in Figure 2.1a, a set of N pairs of real numbers for which we assume that all x_i are pairwise different. The knowledge about this data to be uncovered is the functional dependency between the two data dimensions that were used to generate the data. This is called a regression problem and is strongly under-specified:

Problem 2 (Regression)
Input: A set of N pairs of real numbers $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N \subseteq \mathcal{X} \times \mathcal{Y}$

Output: The functional dependency $f : \mathcal{X} \to \mathcal{Y}$ used to generate the data.

There exist uncountably many functions on the real numbers that explain any finite set of observed points. The problem is therefore not a computational problem and asking for a solution or algorithm for this problem is not a well-posed question.

One can, however, derive computational problems from the regression problem by making assumptions about the nature of the function f that characterize a unique solution. For example, one could ask for the simplest polynomial that explains the data.

Problem 3 (Lagrange Polynomial)
Input: A set of N pairs of real numbers $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N \subseteq \mathcal{X} \times \mathcal{Y}$

Output: The polynomial of smallest degree for which $f(x_n) = y_n$ holds for all $(x_n, y_n) \in \mathcal{D}$.

It can be shown that this is indeed a well-posed computational problem whose unique solution is the Lagrange polynomial L [121] given by the explicit form

$$\begin{aligned} L(x) &= \sum_{i=0}^{N} y_i \ell_i(x), \text{ with} \\ \ell_i &= \prod_{\substack{j=0 \\ j \neq i}}^{N} \frac{x - x_j}{x_i - x_j}. \end{aligned} \tag{2.1}$$

Figure 2.1b shows the Lagrange polynomial interpolating the example dataset. The figure also shows another derived computational problem of linear interpolation. Here, the function is defined as being piecewise linear between the different data points, thus also reproducing the data.

Both of these problems and derived algorithms would typically not be identified as ML approaches, as it can be argued that they do not derive new insights from data.

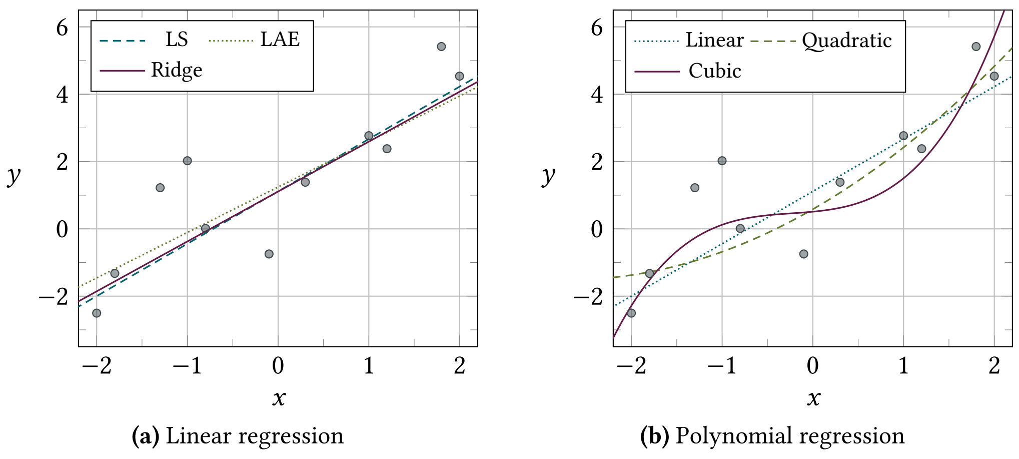


Figure 2.2: Problems in machine learning do not have a well-defined solution but require additional assumptions about the hypothesis space and error function. For the linear regression problem (left), the squared error (LS), absolute error (LAE) and Ridge error functions lead to different solutions. Similarly, alternating the hypothesis space between polynomials of different degrees (right) yields qualitatively different results.

Because of the explicit nature of their algorithms as seen in (2.1), they much more closely resemble classical algorithms such as sorting algorithms.

A common approach to formulating a problem that lets the data speak for itself is to formulate more implicit requirements. Instead of describing exactly one possible function (such as the Lagrange Polynomial), one can choose a broader set of candidate functions or hypotheses $\mathscr{H}$ and then select one of the candidates $f \in \mathscr{H}$ that is optimal with respect to some measure of performance. For example, a common assumption about the process used to generate the data is that it separates into two additive components

$$y_n = f(x_n) + \epsilon(x_n). \tag{2.2}$$

The first summand f captures the truly informative functional dependency between x and y that applies for all observations while the second term ϵ captures local error or noise that can be ignored.

A direct consequence of this assumption is that the output of the regression problem f need no longer interpolate the data perfectly. At the same time, additional reasoning is required about how far from the data f is allowed to be or, equivalently, about the shape of ϵ. One can choose f to be a linear function, giving rise to the linear regression problem.

Problem 4 (Linear regression)
Input: A set of N pairs of real numbers $\mathscr{D} = \{(x_n, y_n)\}_{n=1}^N \subseteq \mathscr{X} \times \mathscr{Y}$ and an error function $e : \mathscr{H} \times \mathscr{X} \times \mathscr{Y} \to \mathbb{R}$

Output: A function f such that

$$f \in \operatorname*{arg\,min}_{f \in \mathscr{H}} e(f, \mathscr{D}) \tag{2.3}$$

with $\mathscr{H}$ being the set of linear functions.

Figure 2.2a shows the different results of three algorithms with different choices of error functions e. With $f(x) = \mathbf{W}x + b$ those are

Squared error: $e(f, \mathscr{D}) = \sum_{n=1}^N (y_n - f(x_n))^2$

Absolute error: $e(f, \mathscr{D}) = \sum_{n=1}^N |y_n - f(x_n)|$

Ridge: $e(f, \mathscr{D}) = \sum_{n=1}^N (y_n - f(x_n))^2 - \|\mathbf{W}\|_2 - b^2$.

Altering the set of hypotheses $\mathscr{H}$ between linear, quadratic and cubic polynomials together with the squared error function results in the functions shown in Figure 2.2b.

It is important to note that none of these proposed algorithms and plotted functions is objectively the correct solution to the regression problem. None of them is equal to the function that was used to generate the data, and even if it was, there would be no way to tell. A core property of machine learning problems is that what characterizes the correct solution is an inherently subjective question. This subjectiveness is represented in multiple choices:

1. The assumed structure underlying the data.
2. The space of hypotheses for valid solutions.
3. The algorithm used to select from these hypotheses.

In the following, statistical learning and Bayesian machine learning are introduced as tools to formalize these choices and establish a mathematical framework.

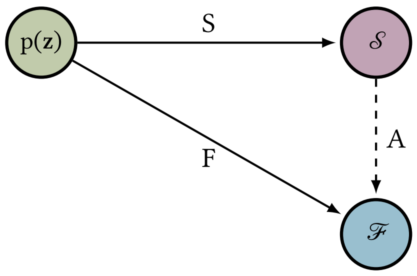

$$\begin{tikzcd}
p(z) \arrow[r, "S"] \arrow[dr, "F"'] & \mathscr{S} \arrow[d, dashed, "A"] \\
& \mathscr{F}
\end{tikzcd}$$

Figure 2.3: Statistical learning characterizes machine learning through a latent data distribution $p(\mathbf{z})$. The task is to learn about a functional of interest $F[p(\mathbf{z})] = f \in \mathscr{F}$ based on a set of observations $S[p(\mathbf{z})] \in \mathscr{S}$. A machine learning algorithm $A : \mathscr{S} \to \mathscr{F}$ is successful if it recovers f such that $A \circ S \simeq F$.

2.2 Statistical Learning

The objective of machine learning is to derive knowledge about a process generating the data from a limited set of observations. Knowledge is represented as a model that both explains existing observations and can generalize to new data points. Statistical learning theory [48, 117] offers a set of tools to reason about generalization and to formulate what it means for a model to be good. This section presents an interpretation of statistical learning theory similar to the definition of probabilistic numerics in [26, 83].

In the following, we denote the space of probability measures over a set $\mathscr{Z}$ as $\mathscr{P}_{\mathscr{Z}}$. We adopt an overloaded notation common in machine learning where $p(\mathbf{z})$ can both refer to a probability measure and the evaluation of the same probability measure on a specific point $\mathbf{z} \in \mathscr{Z}$ depending on the context. Similarly, $\mathbf{z}$ can both refer to a random variable with the distribution $p(\mathbf{z})$ and an element $\mathbf{z} \in \mathscr{Z}$. That is, $\mathbf{k} \sim p(\mathbf{z})$ denotes that the distribution of the random variable $\mathbf{k}$ is $p(\mathbf{z})$ and $\mathbb{E}[\mathbf{z}] = \int \mathbf{z}\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}$ denotes the expected value of the random variable $\mathbf{z}$ under the distribution $p(\mathbf{z})$. In less ambiguous notation one would assume a random variable $\mathbf{Z}$ with distribution $p(\mathbf{Z})$ and denote the expected value as $\mathbb{E}[\mathbf{Z}] = \int_{\mathscr{Z}} z\, p(\mathbf{Z} = z)\, \mathrm{d}z$. Here, $\mathbf{Z}$ and z are identified with each other as $\mathbf{z}$.

Assume an unknown data-generating distribution $p(\mathbf{z}) \in \mathscr{P}_{\mathscr{Z}}$ over a known space of observations $\mathscr{Z}$. The task in a machine learning problem is to infer properties of this distribution given a set of observations $\mathscr{D} \in \mathscr{S}$ obtained via some sampling operator $S : \mathscr{P}_{\mathscr{Z}} \to \mathscr{S}$. A common choice of S is a draw of N independent points $\mathscr{D} = \{\mathbf{z}_1, \dots, \mathbf{z}_N | \mathbf{z}_i \sim p(\mathbf{z})\} \subseteq \mathscr{Z}$ with $\mathscr{S} = \mathscr{Z}^N$. Other choices include S into the learning problem such as in active learning [81] or reinforcement learning [111] settings. Depending on the machine learning problem, the task might not be to identify $p(\mathbf{z})$ as a whole but instead identify some functional $f \in \mathcal{F}$ obtained via the functional operator $F : \mathcal{P}_{\mathcal{Z}} \to \mathcal{F}$. This formulation encompasses a large number of different machine learning problems via different choices for the spaces and operators. We give a few examples here.

Problem 5 (Common machine learning problems)

Parameter estimation In a simple case, the functional f can be chosen to be some constant statistic of $p(\mathbf{z})$ such as the mean

$$F_{\text{mean}}[p(\mathbf{z})] = \int \mathbf{z}\, p(\mathbf{z})\, d\mathbf{z}, \tag{2.4}$$

with $\mathcal{F} = \mathcal{Z}$.

Regression Alternatively, assuming that $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ separates into tuples of inputs $\mathcal{X}$ and continuous outputs $\mathcal{Y}$ and choosing

$$F_{\text{reg}}[p(\mathbf{x}, \mathbf{y})](\mathbf{x}_*) = p(\mathbf{y}_* \mid \mathbf{x}_*) \tag{2.5}$$

with $\mathcal{F} = \mathcal{X} \to \mathcal{P}_{\mathcal{Y}}$ gives rise to the regression problem discussed in Section 2.1.

Classification Classification is closely related to regression and assumes the same separation $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. In contrast to regression, $\mathcal{Y}$ is assumed to be discrete, often without a known order.

Dimensionality Reduction In the dimensionality reduction or manifold learning problem, the assumption is that the support of the data distribution $p(\mathbf{z})$ is a low dimensional manifold in $\mathcal{Z}$. The task is to find a distribution $p(\mathbf{m}) \in \mathcal{P}_{\mathcal{M}}$ and a mapping $f : \mathcal{M} \to \mathcal{Z}$ with

$$\begin{aligned} F_{\text{dim}}[p(\mathbf{z})] &= (p(\mathbf{m}), f), \text{ such that} \\ f(p(\mathbf{m})) &= p(\mathbf{z}) \end{aligned} \tag{2.6}$$

and $\mathcal{F} = \mathcal{P}_{\mathcal{M}} \times (\mathcal{M} \to \mathcal{Z})$ and a space $\mathcal{M}$ of lower dimensionality than $\mathcal{Z}$. If $\mathcal{M} = \mathcal{Z}$ this problem is called density estimation.

A machine learning Algorithm $A : \mathcal{S} \to \mathcal{F}$ is a function mapping from the space of samples to the space of functionals, thereby recovering structure from data. Such an algorithm is successful if it recovers the correct functional $f \in \mathcal{F}$ given observations $\mathcal{D}$.

Figure 2.3 shows the introduced components in a directed diagram. An algorithm is successful if this diagram (approximately) commutes such that $A \circ S \simeq F$. Similarly, a model $f_A \in \mathscr{F}$ can be considered good if it is similar to the true functional f.

The similarity of two functionals can be measured via a distance measure in $\mathscr{F}$. In the case of learning about global properties such as the mean of $\mathrm{p}(\mathbf{z})$, a possible choice is the standard Euclidean norm. Considering the regression case, where $\mathscr{F}$ is a function space mapping inputs to (possibly distributions over) outputs, the effects of choosing a distance measure is more subtle. While it is reasonable to require pointwise similarity, the choice of which points to evaluate allows us to define what we mean by generalization: Besides explaining the observations in $\mathscr{D}$, a good model should yield correct predictions for the complete data distribution $\mathrm{p}(\mathbf{z}) = \mathrm{p}(\mathbf{x}, \mathbf{y})$. The concept of risk minimization is based on this observation.

Definition 6 (Risk minimization in the regression problem)
Given a loss function $\ell : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}$ and a data distribution $\mathrm{p}(\mathbf{x}, \mathbf{y})$ for a regression problem, the *risk* relative to ℓ is defined as

$$\mathrm{R}_\ell : \begin{cases} \mathscr{F} \to \mathbb{R} \\ f_A \mapsto \displaystyle\int \ell(\mathbf{y}, f_A(\mathbf{x}))\, \mathrm{p}(\mathbf{x}, \mathbf{y})\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{y}. \end{cases} \tag{2.7}$$

Risk minimization for a hypothesis space $\mathscr{H} \subseteq \mathscr{F}$ selects a hypothesis $\hat{f}_A$ with the smallest possible risk

$$\hat{f}_A \in \operatorname*{arg\,min}_{f_A \in \mathscr{H}} \mathrm{R}_\ell(f_A). \tag{2.8}$$

A model generalizes with respect to a loss function ℓ if it does not only explain the specific dataset $\mathscr{D}$ well but all possible choices of $\mathscr{D}$ via S. While risk minimization is a theoretical tool to define generalization, it does not immediately yield a learning algorithm. Since it includes an expectation over the unknown data distribution, the risk-term cannot be evaluated directly. Instead, calculating a Monte-Carlo estimate over the training data $\mathscr{D}$ gives rise to a fundamental machine learning algorithm, empirical risk minimization.

Definition 7 (Empirical risk minimization in the regression problem)
Given a loss function $\ell : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}$ and a data distribution $\mathrm{p}(\mathbf{x}, \mathbf{y})$ for a regression

problem, the *empirical risk* relative to ℓ is defined as

$$\mathrm{R}_\ell^{\mathrm{emp}} : \begin{cases} \mathscr{F} \to \mathbb{R} \\ f_A \mapsto \dfrac{1}{N} \displaystyle\sum_{i=1}^{N} \ell(\mathbf{y}, f_A(\mathbf{z})) \end{cases} \tag{2.9}$$

Empirical risk minimization for a hypothesis space $\mathscr{H} \subseteq \mathscr{F}$ selects a hypothesis $\hat{f}_A$ with smallest possible risk

$$\hat{f}_A \in \underset{f_A \in \mathscr{H}}{\arg\min}\, \mathrm{R}_\ell^{\mathrm{emp}}(f_A). \tag{2.10}$$

Intuitively, risk minimization describes the global properties of f_A which get approximated by a number of local properties at the observations in empirical risk minimization. The two questions

1. under which conditions $\mathrm{R}^{\mathrm{emp}}$ converges to R for $N \to \infty$ and
2. if so, what the convergence rates are

underpin (statistical) learning theory [119]. It is safe to assume that for small N, $\mathrm{R}^{\mathrm{emp}}$ can significantly underestimate the true risk as the error on parts of the data distribution is not considered at all. This problem is called overfitting to the available training data, an example of which can arguably be seen in Figure 2.1b.

In practice, if the collection of sufficient data is not possible, overfitting has to be avoided via problem-dependent choices for ℓ and $\mathscr{H}$. Additionally, the empirical risk minimization algorithm is often extended to regularizing loss functions of the form $\ell' : \mathscr{F} \times \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}$ which depend on the structure of the candidate as well as its predictions. The Ridge error function shown in Figure 2.2a is an example of extending the least-squares error function with a preference for parameters with small absolute value. Regularization [83] adds back a global component to empirical risk minimization and has a close relation to the choice of hypothesis space $\mathscr{H}$. The constraint that $f_A \in \mathscr{H} \subseteq \mathscr{F}$ can be thought of as a binary regularization term that adds infinite loss to the set $\mathscr{F} \setminus \mathscr{H}$. Conversely, continuous regularization terms formulate softer and less rigorous preferences within $\mathscr{H}$, for example for structurally simpler solutions [13, 112].

For complex loss functions and hypothesis spaces, finding the true minimum $\hat{f}_A$ is often unfeasible. Another common extension is to modify the optimization scheme to increase the likelihood of selecting a favorable solution. Examples include the usage of test sets or validations sets [13], cross validation [108], early stopping [80] or specific parameter choices [34].

It is often possible to reformulate choices in loss functions as changes in the optimization scheme or constrain a more general hypothesis space with stricter regularization. It is not clear how a machine learning algorithm should be formulated from a formal perspective. However, problem-dependent adjustments to learning algorithms are generally informed by the knowledge provided by domain-experts about the process that generated the data. An additional dimension is therefore given by the need to communicate assumptions and effects of choices with stakeholders that do not have a deep understanding of the field. If a learning algorithm should be interpretable and understandable, assumptions should be explicit, and optimization schemes should be simple. The next section introduces Bayesian machine learning as a rigorous formal framework that builds on statistical learning theory and enables the principled formulation of complex hypothesis spaces.

2.3 Bayesian Machine Learning

The central modeling assumption of statistical learning theory is the unknown data distribution $\mathrm{p}(\mathbf{z})$. Based on a limited number of observations, the task is to learn a model of either the complete distribution or some functional $F[\mathrm{p}(\mathbf{z})]$. Instead of selecting one solution (for example, using some regularization scheme), Bayesian approaches accept the inevitability of uncertainty due to the underspecified ML problem and represent it explicitly. Following the definition of functionals F, a Bayesian model is often formulated through independence assumptions in the data distribution. Independence assumptions induce a factorization of the data distribution. Structural constraints can then be put on the different factors to encode expert knowledge.

Since the available data is not sufficient to identify the unique correct solution, the goal of Bayesian machine learning is to infer a distribution of plausible solutions weighed by their likelihood given the data instead. The posterior distribution of plausible models is a combination of the prior assumptions about the underlying structure and their capability of explaining the observations. This section introduces probabilistic generative models as a natural consequence of the assumptions in statistical learning theory. Probabilistic generative models form a principled way to formulate interpretable hypothesis spaces with carefully chosen assumptions. With Bayes' rule, a structurally simple and formally consistent learning algorithm can be formulated to infer knowledge from data.

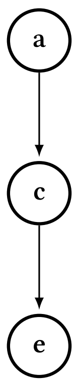
Directed Graphical Models

Any joint probability distribution $\mathrm{p}(\mathbf{a}, \mathbf{c}, \mathbf{e})$ can be formulated as a product of partial distributions via the chain rule of probability [81] as

$$\begin{aligned}\mathrm{p}(\mathbf{a}, \mathbf{c}, \mathbf{e}) &= \mathrm{p}(\mathbf{a} \mid \mathbf{c}, \mathbf{e})\, \mathrm{p}(\mathbf{c} \mid \mathbf{e})\, \mathrm{p}(\mathbf{e}) \\ &= \mathrm{p}(\mathbf{e} \mid \mathbf{a}, \mathbf{c})\, \mathrm{p}(\mathbf{a} \mid \mathbf{c})\, \mathrm{p}(\mathbf{c}) \\ &= \ldots\end{aligned} \tag{2.11}$$

The chain rule is symmetric in the sense that the order of chaining random variables does not matter. All possible interdependencies between random variables are represented in the expansion of conditional probabilities.

A factorization along the chain rule does not impose any structure on the underlying distribution. The first step in Bayesian modeling is to impose such structure by assuming conditional independence between variables. Directed graphical models are a readable visualization of such independence assumptions. A directed graphical model is a directed graph where every random variable is a node $\textcircled{a}$ and an edge $\textcircled{a} \longrightarrow \textcircled{c}$ denotes a dependency of $\mathbf{c}$ on $\mathbf{a}$. In contrast to the factorization according to the chain rule, the graphical model below encodes the assumption that $\mathbf{e}$ is independent of $\mathbf{a}$ given $\mathbf{c}$.



$$\mathrm{p}(\mathbf{a}, \mathbf{c}, \mathbf{e}) = \begin{cases} \mathrm{p}(\mathbf{a}) \\ \cdot\, \mathrm{p}(\mathbf{c} \mid \mathbf{a}) \\ \cdot\, \mathrm{p}(\mathbf{e} \mid \mathbf{c}) \end{cases} \tag{2.12}$$

More formally, the factorization belonging to a graphical model with nodes $\mathbf{a}_1, \ldots, \mathbf{a}_N$ is given by

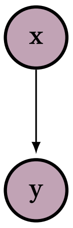
$$\mathrm{p}(\mathbf{a}_1, \ldots, \mathbf{a}_N) = \prod_{n=1}^{N} \mathrm{p}(\mathbf{a}_n \mid \mathrm{parents}(\mathbf{a}_n)) \tag{2.13}$$

with parents($\mathbf{a}_n$) denoting the set of nodes with an edge towards $\mathbf{a}_n$. A cycle $\begin{tikzcd} \textcircled{a} \arrow[r, bend left] & \textcircled{c} \arrow[l, bend left] \end{tikzcd}$ denotes that the variables $\mathbf{a}$ and $\mathbf{c}$ need to be considered jointly as $\mathrm{p}(\mathbf{a}, \mathbf{c})$. Graphical models have been studied in great detail. We refer to [13, 35, 81, 117] for additional information.

Generative Models

In the algorithmic view of machine learning problems formulated in Section 2.1, the observational data is used to formulate an error function to select a model from a set of candidates without the notion of explaining said data. To formulate what it means to generalize, statistical learning theory in Section 2.2 bases the learning problem on an unknown or latent data distribution, with the observational data being just one of many possible samples from that distribution. A generative model captures this sampling process and formulates an algorithm to generate arbitrary data sets from the data distribution $\mathrm{p}(\mathbf{z})$. A generative model is successful if the observational data set is a likely draw from the model. Using the regression problem as an example, we will now introduce the building blocks of a Bayesian generative model.

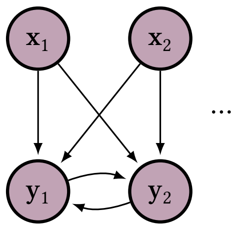
The functional of interest $F_{\text{reg}} = \mathrm{p}(\mathbf{y} \mid \mathbf{x})$ for the regression problem formulated in (2.5) is the conditional probability of the output $\mathbf{y}$ given the input $\mathbf{x}$ for the data distribution $\mathrm{p}(\mathbf{z}) = \mathrm{p}(\mathbf{x}, \mathbf{y})$. Formulating a generative model starts with the factorization of the data distribution such that this conditional is made explicit:



$$\mathrm{p}(\mathbf{x}, \mathbf{y}) = \begin{cases} \mathrm{p}(\mathbf{x}) \\ \cdot\, \mathrm{p}(\mathbf{y} \mid \mathbf{x}) \end{cases} \tag{2.14}$$

Here, the color of the nodes $\textcircled{x}$ and $\textcircled{y}$ indicates that both nodes are part of the data distribution and are directly observed in a data set. We will introduce other nodes below. Closed forms for both $\mathrm{p}(\mathbf{x})$ and $\mathrm{p}(\mathbf{y} \mid \mathbf{x})$ would fully characterize the data distribution. To solve the regression problem, however, only the second term is required.

In the notation typically employed in machine learning, the variables $\mathbf{x}$ and $\mathbf{y}$ in the graphical model denote both the factorization of the generative process as well as the functional dependencies between concrete realizations of $\mathbf{x}$ and $\mathbf{y}$. Assuming a regression problem over the reals $\mathcal{X} = \mathcal{Y} = \mathbb{R}$, the functional $\mathrm{p}(\mathbf{y} \mid \mathbf{x})$ is an infinite-dimensional object. To simplify reasoning about this object, we will reason about finitely many realizations of this process, the pairs $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_N, \mathbf{y}_N)$. This formulation describes both the model search and prediction steps: If the N pairs are all part of some data set $\mathcal{D}$, model-search can be formulated as finding a closed-form for $p(\mathbf{y}_n \mid \mathbf{x}_n)$ that explains the observations well. For prediction, the graphical model is typically augmented with a new pair $(\mathbf{x}_*, \mathbf{y}_*) \notin \mathcal{D}$, an arbitrary but previously unseen point. Making a prediction is equivalent to evaluating the closed form $p(\mathbf{y}_* \mid \mathbf{x}_*)$. To simplify notation, we identify $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ and $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_N)$. Using this formulation, the graphical model is reformulated as



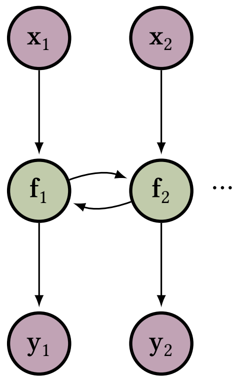
$$p(\mathbf{x}, \mathbf{y}) = \begin{cases} p(\mathbf{x}_1, \mathbf{x}_2) \\ \cdot\, p(\mathbf{y}_1, \mathbf{y}_2 \mid \mathbf{x}_1, \mathbf{x}_2). \end{cases} \tag{2.15}$$

Similar to the assumption of independent draws for a data set $\mathcal{D}$ in statistical learning theory, the inputs $\mathbf{x}_1, \mathbf{x}_2, \dots$ to a generative regression model are typically assumed to be independent. The outputs $\mathbf{y}_1, \mathbf{y}_2, \dots$ are not independent however because they have been generated using the same functional dependency f we wish to model.

In the next step, we expand the model to reflect the fact that the $\mathbf{y}_N$ are conditionally independent given this functional dependency f. The rule of total probability [13] states that for any random variable $\mathbf{f}$, it holds that

$$p(\mathbf{y} \mid \mathbf{x}) = \int p(\mathbf{y} \mid \mathbf{f}, \mathbf{x})\, p(\mathbf{f})\, d\mathbf{f}. \tag{2.16}$$

This rule allows us to add arbitrary nodes to the graphical model and recover the original distribution via marginalization, the calculation of the expectation with respect to the variable that should be removed. Introducing a new variable is helpful because it allows us to formulate independence assumptions explicitly. To achieve conditional independence of the $\mathbf{y}_n$, we add variables $\mathbf{f}_n$ which represent the function values $\mathbf{f}_n = f(\mathbf{x}_n)$. Since the input $\mathbf{x}_n$ is no longer relevant once this function value is known, we assume that $p(\mathbf{y}_n | \mathbf{f}_n, \mathbf{x}_n) = p(\mathbf{y}_n | \mathbf{f}_n)$. This leads to the new graphical model



$$p(\mathbf{x}, \mathbf{y}) = \begin{cases} p(\mathbf{x}_1, \mathbf{x}_2) \\ \cdot \displaystyle\int p(\mathbf{f}_1, \mathbf{f}_2 | \mathbf{x}_1, \mathbf{x}_2) \\ \quad \cdot p(\mathbf{y}_1 | \mathbf{f}_1)\, p(\mathbf{y}_2 | \mathbf{f}_2) \\ \quad \cdot \mathrm{d}\mathbf{f}_1 \, \mathrm{d}\mathbf{f}_2 \end{cases} \tag{2.17}$$

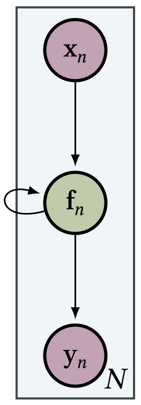
with the additional latent nodes $\textcircled{\mathbf{f}}$ which are never directly observed. The function values are (usually) assumed to not be directly observed due to the additive noise assumption in (2.2), which states that only a noisy version $\mathbf{y}_n = f(\mathbf{x}_n) + \epsilon_n$ of the true functional dependency is observed to motivate regularization terms. In the graphical model above, both terms are represented via

$p(\mathbf{f}_n | \mathbf{x}_n)$: The true functional dependency f, the functional of interest.

$p(\mathbf{y}_n | \mathbf{f}_n)$: The independent noise term ϵ.

In this model, the noise term contains all part of the generative process for $\mathbf{y}_n$ which cannot be explained by $\mathbf{x}_n$ and is assumed to be fully independent given $\mathbf{f}_n$.

The graphical model now contains a chain of nodes $\mathbf{x}_n$, $\mathbf{f}_n$, $\mathbf{y}_n$ that is repeated for every data point. This is a common pattern for generative models. While the functional of interest f changes depending on the position in the input space $\mathbf{x}$, the structure of the generative process is assumed to be the same for all observations. To explicitly represent this repetition, we introduce plate notation:



$$
p(\mathbf{x}, \mathbf{y}) = \begin{cases} p(\mathbf{x}_1, \dots, \mathbf{x}_N) \\ \quad \cdot \int p(\mathbf{f}_1, \dots, \mathbf{f}_N | \mathbf{x}_1, \dots, \mathbf{x}_N) \\ \quad \cdot \prod_{n=1}^{N} p(\mathbf{y}_n | \mathbf{f}_n) \\ \quad \cdot \, d\mathbf{f}_1 \dots d\mathbf{f}_N \end{cases} \tag{2.18}
$$

Within the plate, all nodes indexed with n are repeated N times. The loop on $\mathbf{f}_n$ denotes that all $\mathbf{f}_n$ are dependent on each other. Having separated the functional of interest from the noise term, the next step is to formulate a representation of $p(\mathbf{f}_1, \dots, \mathbf{f}_N | \mathbf{x}_1, \dots, \mathbf{x}_N)$. The choice of this distribution over functions is informed by two challenges. First, the support of the distribution can be restricted to a specific class of functions to shrink the hypothesis space. And second, for a model to be the result of an algorithm, it needs to be represented via finitely many parameters.

There exist two paths to achieving a finite representation:

Non-parametric: The functional dependency is represented implicitly using the existing observations $\mathcal{D}$. Examples include nearest neighbor models, Gaussian processes or polynomial splines.

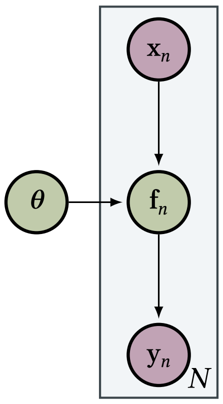
Parametric: The functional dependency is represented explicitly using a specific class of functions with parameters $\boldsymbol{\theta}$. Examples include neural networks, linear regression or polynomial regression.

The model in (2.18) is non-parametric because it does not contain any parameters besides the N pairs $(\mathbf{x}_n, \mathbf{y}_n)$. Because the functional f is implicit, all $\mathbf{f}_n$ inform each other.

If f is represented completely and explicitly via a set of parameters $\boldsymbol{\theta}$ in a parametric model, the $\mathbf{f}_n$ are independent given $\boldsymbol{\theta}$. If this conditional independence

$$
p(\mathbf{f}_n | \boldsymbol{\theta}, \{\mathbf{f}_m | m \in \{1, \dots, N\}, m \neq n\}) = p(\mathbf{f}_n | \boldsymbol{\theta}) \tag{2.19}
$$

holds, $\boldsymbol{\theta}$ is called a sufficient statistic for the functional f. In the parametric graphical model



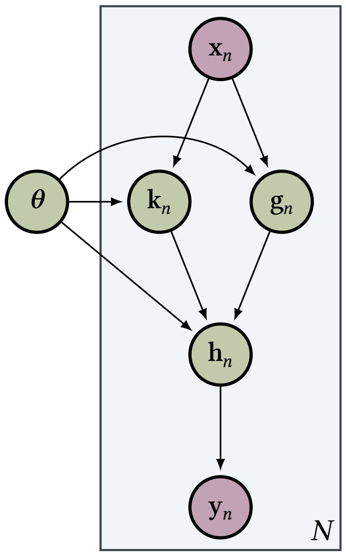
$$\mathrm{p}(\mathbf{x}, \mathbf{y}) = \begin{cases} \mathrm{p}(\mathbf{x}) \\ \cdot \int \prod_{n=1}^{N} \mathrm{p}(\mathbf{f}_n \mid \mathbf{x}_n, \boldsymbol{\theta}) \\ \quad \cdot \mathrm{p}(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta} \\ \quad \cdot \prod_{n=1}^{N} \mathrm{p}(\mathbf{y}_n \mid \mathbf{f}_n) \\ \quad \cdot \mathrm{d}\mathbf{f}_1 \dots \mathrm{d}\mathbf{f}_N \end{cases} \tag{2.20}$$

the joint distribution of the $\mathbf{f}_n$ factorizes given the parameters $\boldsymbol{\theta}$ and the loop no longer exists. Taking linear regression as an example, we assume the functional to be of the shape $f(x) = \mathbf{W}x + \mathbf{b}$ and choose the parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$. The functional dependency in the graphical model is then given by

$$\mathrm{p}(\mathbf{f}_n \mid \mathbf{x}_n, \boldsymbol{\theta}) = \delta(\mathbf{W}\mathbf{x}_n + \mathbf{b}), \tag{2.21}$$

where $\delta(\cdot)$ denotes the Dirac delta distribution [81]. Since $\boldsymbol{\theta}$ is a sufficient statistic, other observations are not required to evaluate the linear function. Even though we made the strong structural assumption about the linearity of the function resulting in a delta distribution, the marginalization of $\mathrm{p}(\boldsymbol{\theta})$ can still lead to more complicated $\mathrm{p}(\mathbf{y} \mid \mathbf{x})$ distributions.

Additional latent structure can be introduced to formulate more complex generative models. Assume, for example, that the functional of interest is known to have the shape $f(x) = h(k(x), g(x))$. The corresponding graphical model



$$
\mathrm{p}(\mathbf{x}, \mathbf{y}) = \begin{cases} \mathrm{p}(\mathbf{x}) \\ \cdot \displaystyle\int \prod_{n=1}^{N} \mathrm{p}(\mathbf{k}_n \,|\, \mathbf{x}_n, \boldsymbol{\theta}) \\ \quad \cdot \displaystyle\prod_{n=1}^{N} \mathrm{p}(\mathbf{g}_n \,|\, \mathbf{x}_n, \boldsymbol{\theta}) \\ \quad \cdot \displaystyle\prod_{n=1}^{N} \mathrm{p}(\mathbf{h}_n \,|\, \mathbf{k}_n, \mathbf{g}_n, \boldsymbol{\theta}) \\ \quad \cdot \mathrm{p}(\boldsymbol{\theta}) \,\mathrm{d}\boldsymbol{\theta} \\ \quad \cdot \displaystyle\prod_{n=1}^{N} \mathrm{p}(\mathbf{y}_n \,|\, \mathbf{k}_n, \mathbf{g}_n, \mathbf{h}_n) \\ \quad \cdot \mathrm{d}\{\mathbf{k}_n, \mathbf{g}_n, \mathbf{h}_n\}_{n=1}^{N} \end{cases} \tag{2.22}
$$

contains separate nodes for the different functions. By constructing a hypothesis space as a combination of multiple sub-hypothesis spaces, generative models can represent detailed assumptions while remaining formally principled through marginalization. Having formulated structural assumptions about how a data set has been generated, the next step is interpret these assumptions in a statistical learning context and connect them with data.

Bayesian Inference

The generative model for linear regression formulated in (2.20) formulates a factorization of the unknown data distribution $\mathrm{p}(\mathbf{x}, \mathbf{y})$. By introducing the parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$ which are sufficient statistics for the functional dependency between $\mathbf{x}$ and $\mathbf{y}$ under our assumptions, learning about the regression functional f has been reduced to learning about finitely many parameters. In other words, the formulation of a generative model yields a formal description of what we mean by linear regression in a statistical learning context. The next step is to derive a learning algorithm that connects this model to observations.

In Section 2.2, a good solution to a learning problem was characterized with the risk minimization algorithm, which demands that any data that could be observed form the data distribution would be well-explained by the solution. In the context of generative models, we can formulate the same idea by demanding that sampling from the true data distribution or from the generative model leads to the same data sets. Or equivalently, that samples drawn from the data distribution have high probability under the generative model and vice versa. Since we do not have access to the full data distribution, we use the available observations as an empirical estimate again.

To make statements about the interaction of the parameters $\boldsymbol{\theta}$ and the observations $\mathscr{D} = \{(\mathbf{x}_n, \mathrm{y}_n)\}_{n=1}^N$, we consider their joint probability distribution $\mathrm{p}(\boldsymbol{\theta}, \mathscr{D})$. Using the chain rule, this distribution can be written as a product

$$\mathrm{p}(\boldsymbol{\theta}, \mathscr{D}) = \mathrm{p}(\mathscr{D} | \boldsymbol{\theta}) \, \mathrm{p}(\boldsymbol{\theta}), \tag{2.23}$$

whose terms are called the likelihood $\mathrm{p}(\mathscr{D} | \boldsymbol{\theta})$ and the prior $\mathrm{p}(\boldsymbol{\theta})$. The structural assumptions formulated via the generative model allow us to directly evaluate the likelihood $\mathrm{p}(\mathscr{D} | \boldsymbol{\theta}) = \mathrm{p}(\mathbf{x}, \mathrm{y} | \boldsymbol{\theta})$. The prior $\mathrm{p}(\boldsymbol{\theta})$ is a distribution over all possible parameters $\boldsymbol{\theta}$ and is part of the joint formulated in the generative model. Applying the chain rule on the joint again yields the posterior

$$\mathrm{p}(\boldsymbol{\theta} | \mathscr{D}) = \frac{\mathrm{p}(\mathscr{D} | \boldsymbol{\theta}) \, \mathrm{p}(\boldsymbol{\theta})}{\mathrm{p}(\mathscr{D})}, \tag{2.24}$$

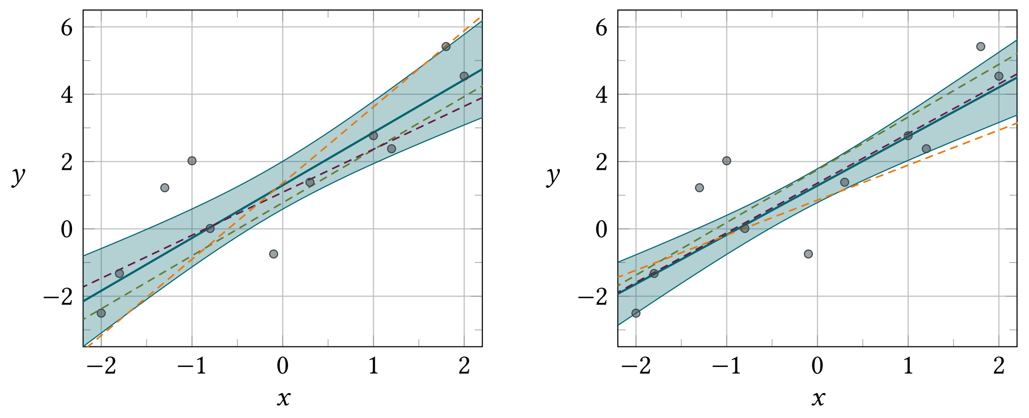
where $\mathrm{p}(\mathscr{D}) = \int \mathrm{p}(\mathscr{D} | \boldsymbol{\theta}) \, \mathrm{p}(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}$ is a combination of the likelihood and and prior terms. Because all terms on the right-hand-side are known, this posterior can be evaluated, yielding a combination of modeling assumptions and observations. If a closed-form solution cannot be found, Bayesian inference is often implemented using approximation techniques such as sampling or variational approaches [13].

Having found a posterior $\mathrm{p}(\hat{\boldsymbol{\theta}}) = \mathrm{p}(\boldsymbol{\theta} | \mathscr{D})$, Bayesian predictions for previously unseen points can be made by replacing $\mathrm{p}(\boldsymbol{\theta})$ with $\mathrm{p}(\hat{\boldsymbol{\theta}})$ in the graphical model. In the regression case, this predictive posterior is given by

$$\mathrm{p}(\mathrm{y}_* | \mathbf{x}_*) = \int \mathrm{p}(\mathrm{y}_* | \mathbf{f}_*) \, \mathrm{p}(\mathbf{f}_* | \mathbf{x}_*, \hat{\boldsymbol{\theta}}) \, \mathrm{p}(\hat{\boldsymbol{\theta}}) \, \mathrm{d}\mathbf{f}_* \, \mathrm{d}\hat{\boldsymbol{\theta}}. \tag{2.25}$$

Because Bayesian linear regression is a parametric model and we assumed that $\boldsymbol{\theta}$ is a sufficient statistic for $\mathbf{f}$, predictions can be made independently of the training data in this case.

In Section 2.2 we argued that considering finitely many observations instead of the full data distribution can lead to overfitting. An overfitted model explains the observed data well but might have a high risk for unseen parts of the data distribution. The empirical risk minimization learning algorithm can be subject to overfitting because



(a) Uncertain prior $\mathbf{W}, \mathbf{b} \sim \mathcal{N}(0, 5^2)$ **(b)** Certain prior $\mathbf{W}, \mathbf{b} \sim \mathcal{N}(1, 1^2)$

Figure 2.4: Two predictive posteriors for a Bayesian linear regression on the same data. Different choices for prior distributions lead to different predictive uncertainties. Which posterior to use is problem-dependent and a subjective choice.

the algorithm selects a single model from the hypothesis space that explains the data well. Bayesian inference is fundamentally different: A Bayesian posterior $p(f | \mathcal{D})$ for a functional of interest f is a distribution over hypotheses rather than one single candidate. This distribution can be thought of as a subset of hypotheses weighed by how well the different hypotheses explain the data. Instead of selecting a good candidate, a Bayesian inference step can be thought of as removing all bad candidates from a hypothesis space. As a consequence, a Bayesian posterior contains both the models in the original hypothesis space that overfit to the observations as well as the more desirable models with similar data likelihoods. Their relative weights in the posterior are dependent on the prior assumptions formulated via the structure of the generative model and the prior $p(\boldsymbol{\theta})$.

Bayesian Model Selection

Statements about Bayesian models are made in terms of probabilities, including predictive posteriors for new inputs $p(\mathbf{y}_* | \mathbf{x}_*, \mathcal{D})$ in a regression problem or a posterior distributions $p(\boldsymbol{\theta} | \mathcal{D})$ for a parameter. This distribution of predictions or parameters can be interpreted as uncertainty due to insufficient knowledge about the generative process. A Bayesian posterior is derived as the combination of the prior knowledge and the data. Since a Bayesian posterior is the combination of the data likelihood and the prior assumptions, the posterior uncertainties are not objective but depend on the prior assumptions.

Figure 2.4 shows two predictive posteriors for a Bayesian linear regression on the data introduced in Figure 2.1a using the graphical model in (2.20). Both models of linear regression are based on the same observations but differ in the prior assumptions about $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$. Choosing a broad prior $\mathbf{W}, \mathbf{b} \sim \mathcal{N}(0, 5^2)$ results in higher posterior uncertainties about both $\mathbf{W}$ and $\mathbf{b}$ than choosing a more narrow prior $\mathbf{W}, \mathbf{b} \sim \mathcal{N}(1, 1^2)$. Samples drawn from the posterior show why this is the case: A broader prior contains linear functions with steep slopes that could explain the data but are never considered by the narrow prior. Not considering these functions might be the desired behavior if implied by the expert knowledge. This situation of having to choose a model from a set of plausible models is a model selection problem.

In principle, model selection is not necessary in Bayesian machine learning. Because the result of inference is a distribution over models weighed by their plausibility, model selection is already included in a Bayesian posterior. However, this is only true for model components that receive a Bayesian treatment by calculating a posterior. No prior is typically placed on the structure of the graphical model itself or the parametric forms of its components, such as the linearity assumption in Bayesian linear regression. For fully Bayesian treatments, priors would also have to be placed on priors, forming hyper-priors. Because such a fully Bayesian treatment is generally computationally intractable, certain parameters or structural assumptions are typically fixed to one specific instantiation, a point estimate. Introducing point estimates also introduces a model selection problem.

Many strategies exist for Bayesian model selection [5, 35, 81] which are not discussed in detail here. Most strategies are related to the ideas of empirical risk minimization and consider a performance measure on the observed data. Given two hypotheses H_1 and H_2, it is common to compare the marginal likelihoods $\mathrm{p}(\mathcal{D} \mid H_1)$ and $\mathrm{p}(\mathcal{D} \mid H_2)$ where $\mathrm{p}(\mathcal{D} \mid H) = \prod_{n=1}^{N} \mathrm{p}(\mathbf{y}_n \mid \mathbf{x}_n, H)$. Choosing the hypothesis H_i with a higher marginal likelihood is called a maximum likelihood estimation. Observing that

$$\frac{\mathrm{p}(H_1 \mid \mathcal{D})}{\mathrm{p}(H_2 \mid \mathcal{D})} = \frac{\mathrm{p}(\mathcal{D} \mid H_1)}{\mathrm{p}(\mathcal{D} \mid H_2)} \frac{\mathrm{p}(H_1)}{\mathrm{p}(H_2)}, \tag{2.26}$$

a common extension to maximum likelihood estimation is to also consider how well the hypotheses conform to the prior assumption through the prior odds $\mathrm{p}(H_1)/\mathrm{p}(H_2)$.

While strategies based on marginal likelihoods often work well in practice, they are limited in scope. Since marginal distributions are considered, models are not evaluated with respect to the latent components in the generative model, such as the shape of $\mathrm{p}(\mathbf{f})$ in Bayesian linear regression. Selecting models using maximum likelihood approaches enforces good predictive uncertainties around the observations in $\mathcal{D}$. No requirements are enforced concerning uncertainties away from the data, the shape of the individual samples drawn from the model or the uncertainties in the latent part of the generative model.

2.4 Gaussian Processes

Bayesian graphical models encode structural assumptions about a machine learning problem. The central assumption in the regression problem is a latent function f that maps inputs to outputs. In the Bayesian linear regression example, a posterior over linear functions could be derived by calculating a posterior over parameters instead, since every parameter explicitly represents a linear function. A Gaussian process (GP) is a non-parametric model that directly represents a distribution over functions [10, 87]. Instead of formulating an explicit parameterized formula, a GP prior encodes more general assumptions about f such as differentiability or variability. A GP posterior can be calculated analytically, making Bayesian inference over GPs computationally feasible.

Gaussian processes are a generalization of the Gaussian distribution to function spaces. A multivariate Gaussian $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ describes a distribution over the finitely many elements in the vector $\mathbf{x}$. Every such element $\mathbf{x}_i$ is normally distributed according to $\mathbf{x}_i \sim \mathcal{N}(\mu_i, \Sigma_{ii})$ and every linear combination of the $\mathbf{x}_i$ is also normally distributed [6]. For every pair $(\mathbf{x}_i, \mathbf{x}_j)$, their covariance is given by $\operatorname{cov}[\mathbf{x}_i, \mathbf{x}_j] = \Sigma_{ij}$. The probability density of $\mathbf{x}$ is given by

$$\begin{aligned} \mathrm{p}(\mathbf{x}) &= \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \Sigma) \\ &= \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right). \end{aligned} \tag{2.27}$$

Multivariate Gaussians have several convenient closure properties [6]. Assume a split $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ into two partial vectors and denote

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}\right) \tag{2.28}$$

the same split of the mean vector $\boldsymbol{\mu}$ and covariance matrix Σ. Then, the marginal distribution of $\mathbf{x}_1$ is also a Gaussian with

$$\begin{aligned} \mathrm{p}(\mathbf{x}_1) &= \int \mathrm{p}(\mathbf{x}_1, \mathbf{x}_2)\,\mathrm{d}\mathbf{x}_2 \\ &= \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1) \end{aligned} \tag{2.29}$$

and the conditional of $\mathbf{x}_1$ given $\mathbf{x}_2$ is a Gaussian as well with

$$\begin{aligned} \mathrm{p}(\mathbf{x}_1 \mid \mathbf{x}_2) &= \frac{\mathrm{p}(\mathbf{x}_1, \mathbf{x}_2)}{\mathrm{p}(\mathbf{x}_2)} \\ &= \mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\Sigma}), \text{ and} \\ \hat{\boldsymbol{\mu}} &= \boldsymbol{\mu}_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2) \\ \hat{\Sigma} &= \Sigma_{11} - \Sigma_{21}\Sigma_{22}^{-1}\Sigma_{12} \\ &= \Sigma_{11} - \Sigma_{12}^{\mathrm{T}}\Sigma_{22}^{-1}\Sigma_{12}. \end{aligned} \tag{2.30}$$

Modeling functions over infinite sets requires an infinite number of random variables, one for every function value. Such structure of a number of possibly dependent random variables mapping from the same probability space to the same value space is called a stochastic process and is represented via a function.

Definition 8 (Stochastic Process)
Given a probability space $(\Omega, \mathcal{F}, P)$, an index set T and a measurable space Y, a stochastic process $\mathbf{X}$ is a function

$$\mathbf{X} : \begin{cases} T \times \Omega \to Y \\ (t, \omega) \mapsto \mathbf{X}_t(\omega) \end{cases} \tag{2.31}$$

mapping indices t to Y-valued random-variables. For a fixed $\omega \in \Omega$, $\mathbf{X}(\cdot, \omega)$ is called a trajectory of the process [6].

The index set of a stochastic process can be an arbitrary set. It is often interpreted as a time index which can be both discrete and continuous. A Gaussian process is a particular stochastic process.

Definition 9 (Gaussian Process)
A stochastic process $\mathbf{X}$ is called a Gaussian process if for any finite subset $\tau \subseteq T$ of its index set, the random variables $\mathbf{X}_\tau$ have a joint Gaussian distribution [6].

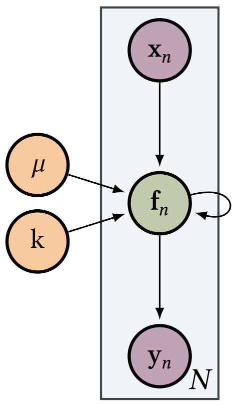


Figure 2.5: The graphical model of a Gaussian process with N observations $(\mathbf{x}_n, \mathbf{y}_n)$. Different observations are independent given the latent function values $\mathbf{f}_n$ which are all jointly Gaussian. They are informed by the mean function μ and kernel k.

When using a Gaussian process $\mathbf{X}$ to model a function $f\colon\ A \to B$, the index set T is assumed to be A and all random variables are B-valued. The random variable $\mathbf{X}_a$ then models the function value $f(a)$ for all $a \in A$. Sampling a trajectory from $\mathbf{X}$ corresponds to sampling one possible function f^*.

Similar to the finite case, the random variables share a dependency structure. Instead of a mean vector $\boldsymbol{\mu}$ and a covariance matrix Σ, a Gaussian process is completely determined by a mean function $\mu(a) = \mathbb{E}[f(a)]$ and a covariance function

$$\begin{aligned} \mathrm{k}(a, a') &:= \mathbb{E}[(f(a) - \mu(a))(f(a') - \mu_f(a'))] \\ &= \operatorname{cov}[f(a), f(a')] \\ &= \operatorname{cov}[\mathbf{X}_a, \mathbf{X}_{a'}] \end{aligned} \tag{2.32}$$

with $a, a' \in A$. The mean function encodes the point-wise mean over all trajectories that could be sampled from $\mathbf{X}$. The covariance function is also called a kernel and describes the interaction between different parts of the function. A function that is distributed according to a Gaussian process is denoted as $f \sim \mathcal{GP}(\mu, \mathrm{k})$.

A GP can be used as a distribution over functions in the graphical model for regression problems in (2.18). Because the random variables $\mathbf{f}_n$ modeling the function value $f(\mathbf{x}_n)$ are jointly Gaussian, they are not independent and thus are connected in the graphical model in Figure 2.5. The choice of the mean function μ and the kernel k describe the GP prior and are often referred to as hyper-parameters.

For convenience, the prior mean function μ is often assumed to be constant zero. This assumption is without loss of generality [87] since otherwise, the observations $(\mathbf{X}, \mathbf{y})$ can be transformed to $\mathbf{y}' = \mathbf{y} - \mu(\mathbf{X})$. The Gaussian process based on the observations $(\mathbf{X}, \mathbf{y}')$ then only models the differences to the mean function. It is the covariance functions that encode the assumptions about the underlying function.

Kernels

The covariance for any pair of random variables $(\mathrm{X}_i, \mathrm{X}_j)$ in a GP is given by the kernel $\operatorname{cov}[\mathrm{X}_i, \mathrm{X}_j] = \mathrm{k}(i, j)$. A kernel, therefore, can not be any arbitrary function but must yield valid covariance matrices Σ. The matrix obtained by applying a kernel pairwise to finitely many random variables is called the Gram matrix. Given two sets $\mathbf{A} = \{\mathbf{a}_i \,|\, i \in [n]\}$ and $\mathbf{B} = \left\{\mathbf{b}_j \,\middle|\, j \in [m]\right\}$ and $[n] = \{1, \ldots, n\}$, the Gram matrix with respect to $\mathbf{A}$ and $\mathbf{B}$ using kernel k is given by

$$\mathrm{k}(\mathbf{A}, \mathbf{B}) = \mathbf{K}_{\mathbf{AB}} := \left(\mathrm{k}(\mathbf{a}_i, \mathbf{b}_j)\right)_{\substack{i \in [n], \\ j \in [m]}}. \tag{2.33}$$

For the Gram matrix to be a valid covariance matrix Σ of a Gaussian distribution, it must be positive definite. Kernels are functions that fulfill the property that for every possible subset of random variables, or more generally every set of elements in their domain, their induced Gram matrix is positive definite.

Definition 10 (Kernel)
Given a non-empty set A, a function

$$\mathrm{k}: \; A^2 \to \mathbb{R} \tag{2.34}$$

is called a (positive definite) kernel or covariance function, if for any finite subset $X \subseteq A$, the Gram matrix $\mathrm{k}(X, X)$ is positive definite.

The kernel is crucial in encoding the assumptions about the function a Gaussian process should estimate. It is a measure of how much different points in the GP's domain inform each other. A natural assumption to make is that the closer together in the domain two points lie, the more similar their function values will be. Similarly, to predict a test point, training points close to it are probably more informative than those further away.

But closeness is not the only possible reason two points could be similar. Assume a function that is a possibly noisy sinusoidal wave with a known frequency. Then, two points that are a multiple of wavelengths apart should also have similar function values. Such a kernel which is not only dependent on the distance between two points but also their position in the input space is called non-stationary. A simple example of such a non-stationary kernel is the linear kernel.

Definition 11 (Linear Kernel)
For a finite dimensional euclidean vector space $\mathbb{R}^d$, the linear kernel is defined as

$$\mathrm{k}_{\text{linear}}(\mathbf{x}_i, \mathbf{x}_j) \coloneqq \mathbf{x}_i^\mathsf{T} \mathbf{x}_j = \langle \mathbf{x}, \mathbf{x}_j \rangle . \tag{2.35}$$

Consider a function $f\colon \mathbb{R} \to \mathbb{R}$ which is distributed according to a GP with the linear kernel $f \sim \mathcal{GP}(\mathbf{0}, \mathrm{k}_{\text{linear}})$. According to the definition of GPs, for any two input numbers $x_i, x_j \in \mathbb{R}$ their corresponding random variables $\mathbf{f}_{x_i}$ and $\mathbf{f}_{x_j}$ have a joint Gaussian distribution

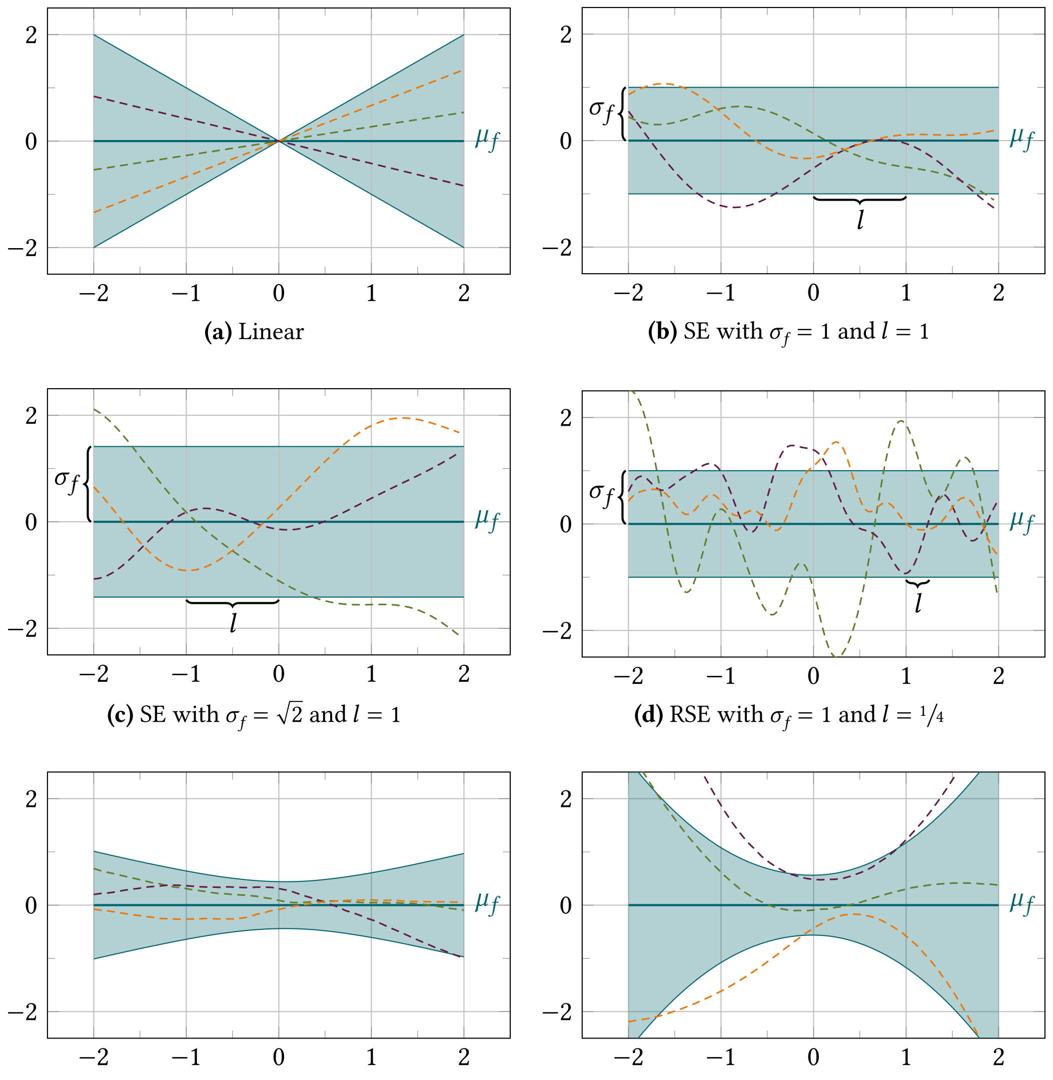
$$\begin{pmatrix} \mathbf{f}_{x_i} \\ \mathbf{f}_{x_j} \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathrm{k}(x_i, x_i) & \mathrm{k}(x_i, x_j) \\ \mathrm{k}(x_j, x_i) & \mathrm{k}(x_j, x_j) \end{bmatrix}\right). \tag{2.36}$$

Assuming that both x_i and x_j are not equal to zero, the correlation coefficient ρ of these two variables is given by

$$\begin{aligned} \rho[\mathbf{f}_{x_i}, \mathbf{f}_{x_j}] &= \frac{\operatorname{cov}[\mathbf{f}_{x_i}, \mathbf{f}_{x_j}]}{\sqrt{\operatorname{var}[\mathbf{f}_{x_i}]}\sqrt{\operatorname{var}[\mathbf{f}_{x_j}]}} \\ &= \frac{\mathrm{k}(x_i, x_j)}{\sqrt{\mathrm{k}(x_i, x_i)}\sqrt{\mathrm{k}(x_j, x_j)}} = \frac{x_i x_j}{\sqrt{x_i^2}\sqrt{x_j^2}} \in \{-1, 1\} . \end{aligned} \tag{2.37}$$

A correlation coefficient of plus or minus one implies that the value of one of the random variables is a linear function of the other. Any function drawn from this Gaussian process, such as the ones shown in Figure 2.6a, is, therefore, a linear function. This observation generalizes to higher dimensions [87]. Gaussian process regression with a linear kernel is equivalent to Bayesian linear regression as discussed in Section 2.3.

It is often of interest to also represent non-linear dependencies. A common approach is to restrict information to local neighborhoods. A kernel which is a function of $\|\mathbf{x}_i - \mathbf{x}_j\|$ is called stationary and is invariant to translations in the input space. The most important stationary kernel is the squared exponential kernel.



(a) Linear

(b) SE with $\sigma_f = 1$ and $l = 1$

(c) SE with $\sigma_f = \sqrt{2}$ and $l = 1$

(d) RSE with $\sigma_f = 1$ and $l = 1/4$

(e) Arc-cos with order 1

(f) Arc-cos with order 2

Figure 2.6: A comparison of samples drawn from GP priors with different kernels and hyperparameters. Dashed lines are single samples and the shaded area depicts two standard deviations around the mean. While samples from the linear kernel are always linear functions, samples from squared exponential (SE) kernels are smooth functions of different variability. The arc cosine kernel mimics the behavior of an infinitely wide neural network with a single hidden layer.

Definition 12 (Squared Exponential Kernel)
For a finite dimensional euclidean vector space $\mathbb{R}^d$, the squared exponential kernel or RBF kernel is defined as

$$\mathrm{k}_{\mathrm{SE}}(\mathbf{x}_i, \mathbf{x}_j) := \sigma_f^2 \cdot \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^{\mathsf{T}} \mathbf{\Lambda}^{-1} (\mathbf{x}_i - \mathbf{x}_j)\right). \tag{2.38}$$

The parameter $\sigma_f^2 \in \mathbb{R}_{>0}$ is called the signal variance and $\mathbf{\Lambda} = \operatorname{diag}(l_1^2, \dots, l_d^2)$ is a diagonal matrix of the squared length scales $l_i \in \mathbb{R}_{>0}$.

The similarity of two data points approaches one when they are close together and for larger distances approaches zero with exponential drop off. It can be shown that this kernel represents all infinitely differentiable functions [87]. Gaussian processes with this covariance function are universal function approximators.

The squared exponential kernel is dependent on multiple parameters that influence its behavior. In contrast to weight parameters in linear regression or constants in physical models, these parameters do not specify the estimated function but rather the prior belief about this function and are therefore hyper-parameters.

The hyper-parameters of the RBF kernel describe the expected dynamic range of the function. The signal variance σ_f^2 specifies the average distance of function values from the mean function. The different length scale parameters l_i roughly specify the distance of data points along their respective axis required for the function values to change considerably. Figure 2.6 compares sample functions drawn from Gaussian processes with the linear kernel, squared exponential kernels with different hyper-parameters, and the arc cosine kernel [22]. Arc cosine kernels mimic the behavior of infinitely wide neural networks with a single hidden layer. The order of an arc cosine kernel specifies the activation function of such a neural network. The first three orders assume step-functions, rectified linear units, or rectified quadratic units respectively. Arc cosine kernels of low order behave similarly to squared exponential kernels in practice but have different generalization properties. While GPs with an RBF kernel return to the prior away from data yielding predictions roughly in the area $\mu_f \pm 2\sigma_f$, the arc cosine kernel of order one generalizes linearly using the derivative at the closest data points.

Predictions and Posterior

To use Gaussian processes for regression, observations need to be combined with a Gaussian process prior $f \sim \mathcal{GP}(\mathbf{0}, \mathrm{k})$ to obtain a predictive posterior. We assume

Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ with $\mathrm{y}_n = f(\mathbf{x}_n) + \epsilon$ and denote the N observations as $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ and $\mathbf{y} = (\mathrm{y}_1, \dots, \mathrm{y}_N)$. The likelihood of the observations given the latent function values $\mathbf{f}$ is given by

$$\begin{aligned} \mathrm{p}(\mathbf{y} \mid f, \mathbf{X}, \sigma) = \mathrm{p}(\mathbf{y} \mid \mathbf{f}, \sigma) &= \prod_{n=1}^{N} \mathcal{N}(\mathrm{y}_n \mid \mathbf{f}_n, \sigma^2) \\ &= \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \sigma^2 \mathrm{I}). \end{aligned} \tag{2.39}$$

Given a vector of hyper-parameters $\boldsymbol{\theta}$, the definition of Gaussian processes yields a joint Gaussian distribution for the latent function values $\mathbf{f}$ given by

$$\mathrm{p}(\mathbf{f} \mid \mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{f} \mid \mathbf{0}, \mathbf{K}_{\mathbf{ff}}) \tag{2.40}$$

where $\mathbf{K}_{\mathbf{ff}} = \mathrm{k}(\mathbf{X}, \mathbf{X})$ denotes the Gram matrix of the observed data. Combining the two distributions according to the law of total probability yields the probability distribution of the outputs conditioned on the inputs and is given by

$$\begin{aligned} \mathrm{p}(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) &= \int \mathrm{p}(\mathbf{y} \mid \mathbf{f})\, \mathrm{p}(\mathbf{f} \mid \mathbf{X}, \boldsymbol{\theta}) \, \mathrm{d}\mathbf{f} \\ &= \int \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \sigma^2 \mathrm{I})\, \mathcal{N}(\mathbf{f} \mid \mathbf{0}, \mathbf{K}_{\mathbf{ff}}) \, \mathrm{d}\mathbf{f} \\ &= \mathcal{N}(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_{\mathbf{ff}} + \sigma^2 \mathrm{I}). \end{aligned} \tag{2.41}$$

Note that this distribution is obtained by integrating over all possible latent function values $\mathbf{f}$ and thereby taking all possible function realizations into account. The closed-form solution of the integral is obtained using well-known results about Gaussian distributions, wich are, for example, detailed in [84].

Now consider a set of new points $\mathbf{X}_*$ for which the predictive posterior should be obtained. By definition of GPs, the latent function values $\mathbf{f}$ of the data set and the latent function values at the new points $\mathbf{f}_* = f(\mathbf{X}_*)$ have the joint Gaussian distribution

$$\mathrm{p}\left(\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \middle| \mathbf{X}, \mathbf{X}_*, \boldsymbol{\theta}\right) = \mathcal{N}\left(\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \middle| \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{ff}} & \mathbf{K}_{\mathbf{f}*} \\ \mathbf{K}_{*\mathbf{f}} & \mathbf{K}_{**} \end{bmatrix}\right). \tag{2.42}$$

Adding the noise model to this distribution leads to the joint Gaussian of training outputs $\mathbf{y}$ and test outputs $\mathbf{f}_*$ which is given by

$$\mathrm{p}\left(\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \middle| \mathbf{X}, \mathbf{X}_*, \boldsymbol{\theta}\right) = \mathcal{N}\left(\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix} \middle| \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{ff}} + \sigma^2 \mathrm{I} & \mathbf{K}_{\mathbf{f}*} \\ \mathbf{K}_{*\mathbf{f}} & \mathbf{K}_{**} \end{bmatrix}\right). \tag{2.43}$$

In this distribution, the training outputs $\mathbf{y}$ are known. The predictive posterior for the test outputs $\mathbf{f}_*$ can therefore be directly obtained by applying (2.30) to (2.43) and is also a Gaussian.

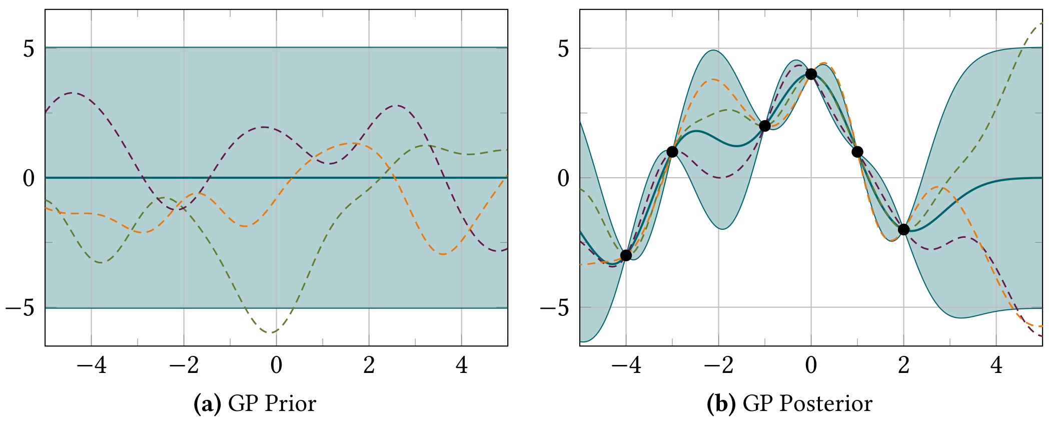


Figure 2.7: A GP prior with an RBF kernel (left) and posterior given a six observations (right). The posterior has a non-zero mean function that interpolates the data exactly. Samples from the posterior interpolate the data but vary in-between.

Lemma 13 (GP predictive posterior)
Assume a latent function with a Gaussian process distribution $f \sim \mathcal{GP}(\mathbf{0}, \mathrm{k})$ and N training points $\mathbf{X}$ with noisy observations of the form $\mathbf{y} = f(\mathbf{X}) + \mathcal{N}(\mathbf{0}, \sigma^2 \mathrm{I})$. The predictive posterior $\mathbf{f}_*$ of the test points $\mathbf{X}_*$ is then given by

$$\begin{aligned} \mathrm{p}(\mathbf{f}_* \,|\, \mathbf{X}, \mathbf{y}, \mathbf{X}_*) &= \mathcal{N}(\mathbf{f}_* \,|\, \boldsymbol{\mu}_*, \Sigma_*), \text{ where} \\ \boldsymbol{\mu}_* &= \mathbf{K}_{*\mathbf{f}} \left(\mathbf{K}_{\mathbf{f}\mathbf{f}} + \sigma^2 \mathrm{I}\right)^{-1} \mathbf{y} \\ \Sigma_* &= \mathbf{K}_{**} - \mathbf{K}_{*\mathbf{f}} \left(\mathbf{K}_{\mathbf{f}\mathbf{f}} + \sigma^2 \mathrm{I}\right)^{-1} \mathbf{K}_{\mathbf{f}*}. \end{aligned} \tag{2.44}$$

This predictive posterior makes it possible to evaluate the function approximation based on the input at arbitrary points in the input space. Since any set of these points always has a joint Gaussian distribution, the predictive posterior defines a new Gaussian process, the posterior Gaussian process, given the observations. This posterior process $\mathcal{GP}(\mu_{\text{post}}, \mathrm{k}_{\text{post}})$ has new mean and covariance functions given by

$$\begin{aligned} \mu_{\text{post}}(\mathbf{a}) &= \mathrm{k}(\mathbf{a}, \mathbf{X}) \left(\mathbf{K}_{\mathbf{f}\mathbf{f}} + \sigma^2 \mathrm{I}\right)^{-1} \mathbf{y} \\ \mathrm{k}_{\text{post}}(\mathbf{a}, \mathbf{b}) &= \mathrm{k}(\mathbf{a}, \mathbf{b}) - \mathrm{k}(\mathbf{a}, \mathbf{X}) \left(\mathbf{K}_{\mathbf{f}\mathbf{f}} + \sigma^2 \mathrm{I}\right)^{-1} \mathrm{k}(\mathbf{X}, \mathbf{b}). \end{aligned} \tag{2.45}$$

Note that the posterior mean function is not necessarily the constant zero function. Figure 2.7 shows samples from a pair of prior and posterior Gaussian processes with an RBF kernel.

Computing the inverse $\left(\mathbf{K_{ff}} + \sigma^2\mathrm{I}\right)^{-1}$ takes $\mathcal{O}(N^3)$ time but can be done as a preprocessing step since it is independent of the test points. Predicting the mean function value of a single test point is a weighted sum of N basis functions $\mu_* = \mathbf{K_{*f}}\boldsymbol{\beta}$ where $\boldsymbol{\beta} = \left(\mathbf{K_{ff}} + \sigma^2\mathrm{I}\right)^{-1}\mathrm{y}$ which can be precomputed. After this pre-computation, predicting the mean of a single test point takes $\mathcal{O}(N)$ time. To predict the variance, it is still necessary to perform a vector-matrix multiplication, which takes $\mathcal{O}(N^2)$ time for a single prediction. Since all of these operations are dependent on the number of training points, evaluating Gaussian processes on large data sets can be computationally expensive. Before introducing sparse approximations with better asymptotic complexity, we first consider how to choose good values for the hyper-parameters $\boldsymbol{\theta}$.

Choosing Hyper-Parameters

In the previous section, we derived the posterior GP given constant hyper-parameters $\boldsymbol{\theta}$. In this case, Gaussian process models do not have to be trained or optimized at all as the posterior GP can be computed analytically. Usually, however, the correct choice of hyper-parameters is not clear a priori. In a fully Bayesian setup we place a prior on the hyper-parameters $\mathrm{p}(\boldsymbol{\theta})$ and marginalize it to derive the dependent distributions

$$\begin{aligned} \mathrm{p}(f) &= \int \mathrm{p}(f \mid \boldsymbol{\theta})\,\mathrm{p}(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \\ \mathrm{p}(\mathrm{y} \mid \mathrm{X}) &= \int \mathrm{p}(\mathrm{y} \mid \mathbf{f})\,\mathrm{p}(\mathbf{f} \mid \mathrm{X}, \boldsymbol{\theta})\,\mathrm{p}(\boldsymbol{\theta})\,\mathrm{d}\mathbf{f}\,\mathrm{d}\boldsymbol{\theta}. \end{aligned} \tag{2.46}$$

Updating the belief about the distribution of the hyper-parameters then becomes part of calculating the posterior using Bayes' theorem. However, the integration required in (2.46) is expensive as no closed form solution exists. While true posteriors can be obtained for GPs with few observations such as in Bayesian optimization or probabilistic numerics contexts [83, 96], the required calculations are often not tractable for larger problems.

A common approximation is to use maximum-a-posteriori point-estimates instead. These estimates are obtained by maximizing $\mathrm{p}(\boldsymbol{\theta} \mid \mathrm{X}, \mathrm{y})$. This posterior is proportional to the numerator of Bayes' theorem and given by

$$\begin{aligned} \mathrm{p}(\boldsymbol{\theta} \mid \mathrm{X}, \mathrm{y}) &\propto \mathrm{p}(\boldsymbol{\theta})\,\mathrm{p}(\mathrm{y} \mid \mathrm{X}, \boldsymbol{\theta}) \\ &= \int \mathrm{p}(\boldsymbol{\theta})\,\mathrm{p}(\mathrm{y} \mid \mathbf{f}, \boldsymbol{\theta})\,\mathrm{p}(\mathbf{f} \mid \mathrm{X}, \boldsymbol{\theta})\,\mathrm{d}\mathbf{f}. \end{aligned} \tag{2.47}$$

If $\mathrm{p}(\boldsymbol{\theta})$ is set to a flat distribution, the prior term vanishes and only the likelihood term remains. Choosing hyper-parameters by maximizing the likelihood term is called a type II maximum likelihood estimate.

The marginal likelihood is the integral of the product of Gaussians in (2.41) given by

$$\mathrm{p}(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{y} \mid \mathbf{0}, \mathbf{K_{ff}} + \sigma^2 \mathrm{I}). \tag{2.48}$$

Instead of maximizing the marginal likelihood directly, it is numerically convenient to minimize the negative logarithm of the likelihood

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= -\log \mathrm{p}(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) \\ &= \frac{1}{2}\mathbf{y}^\top \left(\mathbf{K_{ff}} + \sigma^2 \mathrm{I}\right)^{-1} \mathbf{y} + \frac{1}{2}\log\left|\mathbf{K_{ff}} + \sigma^2 \mathrm{I}\right| + \frac{N}{2}\log(2\pi). \end{aligned} \tag{2.49}$$

Since the logarithm is a monotonous function it does not change the position of optima. The maximum likelihood estimate is the solution of the optimization problem

$$\boldsymbol{\theta}^* \in \operatorname*{arg\,min}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \tag{2.50}$$

and is calculated using standard approaches to non-convex optimization. The computational complexity of evaluating the likelihood term and its derivatives is dominated by the inversion of $\mathbf{K_{ff}} + \sigma^2 \mathrm{I}$ with a time complexity of $\mathcal{O}(N^3)$.

2.5 Sparse Gaussian Processes with Inducing Points

A drawback of Gaussian processes in real-world applications is their high computational cost for large data sets. Assume a data set $(\mathbf{X}, \mathbf{y})$ with N training samples, then the operations on a posterior Gaussian process are usually dominated by the inversion of the kernel matrix $\mathbf{K_{ff}}$ which takes $\mathcal{O}(N^3)$ time. While this inversion can be precomputed, the cost of predicting the mean and variance of one test point remains $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$, respectively. Additionally, these operations have a space requirement of $\mathcal{O}(N^2)$. The goal of sparse approximations of Gaussian processes is to find model representations that avoid the cubic complexities or at least restrict them to the training phase of finding hyper-parameters. This section introduces one type of approximation based on representing the complete data set through a smaller set of points [103]. The next section will place this approximation in a principled variational context [56, 113].

The most direct approach to reducing the computational cost of inverting $\mathbf{K_{ff}}$ is to restrict observations to a small subset of size $M \ll N$ of the original data. Calculating the posterior GP relative to these M observations only has a time complexity of $\mathcal{O}(M^3)$. This approach can work for data sets with a very high level of redundancy but does impose the problem of choosing an appropriate subset. While choosing a random subset can be effective [103], the optimal choice is dependent on the hyper-parameters. Both the subset and the hyper-parameters should, therefore, be chosen in a joint optimization scheme. The selection of an appropriate subset defines a combinatorial optimization problem and is very hard to solve.

To overcome this problem, inducing observation approximations lift the restriction of choosing points from the data set and instead allow arbitrary positions in the input space. The original data set is replaced by an inducing data set $(\mathbf{Z}, \mathbf{u})$ of inducing inputs $\mathbf{Z}$ and inducing variables $\mathbf{u} = f(\mathbf{Z})$ which are equal to the true latent values of the function $f \sim \mathcal{GP}(\mathbf{0}, \mathrm{k})$. Since they are not true observations, they are assumed to be noise-free. Given an inducing data set and hyper-parameters $\boldsymbol{\theta}$, the predictive posterior of this approximation is a standard GP posterior

$$\mathrm{p}(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{Z}, \mathbf{u}, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{K}_{*\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{K}_{**} - \mathbf{K}_{*\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{u}*}) \tag{2.51}$$

with $\mathbf{K_{uu}} = \mathrm{k}(\mathbf{Z}, \mathbf{Z})$ denoting the Gram matrix of the inducing inputs.

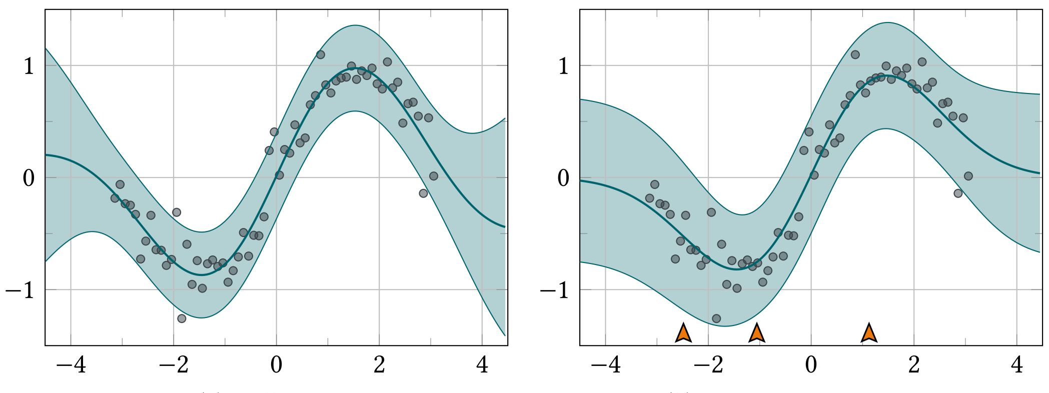
The true data set is independent given the latent function f and can be assumed independent given the inducing data set if it represents f well. The likelihood of the original data under the Gaussian process trained on the inducing data set is given by

$$\begin{aligned}\mathrm{p}(\mathbf{y} \mid \mathbf{X}, \mathbf{Z}, \mathbf{u}, \boldsymbol{\theta}) &= \prod_{n=1}^{N} \mathrm{p}(\mathrm{y}_n \mid \mathbf{x}_n, \mathbf{Z}, \mathbf{u}, \boldsymbol{\theta}) \\ &= \prod_{n=1}^{N} \mathcal{N}\left(\mathrm{y}_n \,\middle|\, \mathbf{K}_{\mathbf{f}_n\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \mathbf{K}_{\mathbf{f}_n\mathbf{f}_n} - \mathbf{K}_{\mathbf{f}_n\mathbf{u}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K}_{\mathbf{u}\mathbf{f}_n} + \sigma^2\right) \\ &= \mathcal{N}\left(\mathbf{y} \,\middle|\, \mathbf{K_{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \operatorname{diag}(\mathbf{K_{ff}} - \mathbf{K_{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K_{uf}}) + \sigma^2\mathrm{I}\right) \\ &= \mathcal{N}\left(\mathbf{y} \,\middle|\, \mathbf{K_{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{u}, \operatorname{diag}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) + \sigma^2\mathrm{I}\right)\end{aligned} \tag{2.52}$$

with $\mathbf{Q_{ff}} \coloneqq \mathbf{K_{fu}}\mathbf{K}_{\mathbf{uu}}^{-1}\mathbf{K_{uf}}$.

Since the inducing variables $\mathbf{u}$ are latent function values, the original GP prior for f is a reasonable prior for their values given by

$$\mathrm{p}(\mathbf{u} \mid \mathbf{Z}) = \mathcal{N}(\mathbf{u} \mid \mathbf{0}, \mathbf{K_{uu}}). \tag{2.53}$$



(a) Full GP (b) FITC approximation

Figure 2.8: A FITC GP approximation compared to a full GP posterior with data from a noisy sine function. The inducing inputs are located at the dart positions. Since the inducing outputs are marginalized, only their *x*-coordinate is meaningful. Three inducing inputs are enough to approximate the full GP with reasonable accuracy.

Using this prior, the inducing variables can be marginalized through an integral of a product of Gaussians in

$$\begin{aligned} \mathrm{p}(\mathbf{y} \mid \mathbf{X}, \mathbf{Z}) &= \int \mathrm{p}(\mathbf{y} \mid \mathbf{X}, \mathbf{Z}, \mathbf{u})\, \mathrm{p}(\mathbf{u} \mid \mathbf{Z})\, \mathrm{d}\mathbf{u} \\ &= \int \mathrm{p}(\mathbf{y} \mid \mathbf{X}, \mathbf{Z}, \mathbf{u})\, \mathcal{N}(\mathbf{u} \mid \mathbf{0}, \mathbf{K}_{\mathbf{uu}})\, \mathrm{d}\mathbf{u} \\ &= \mathcal{N}(\mathbf{y} \mid \mathbf{0}, \mathbf{Q}_{\mathbf{ff}} + \operatorname{diag}(\mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}})), \end{aligned} \tag{2.54}$$

dropping the conditioning on $\boldsymbol{\theta}$ for notational simplicity. Due to the conditional independence assumption of the data given the inducing variables, this formulation is called the fully independent training conditional (FITC) and was introduced by Snelson et al. [101, 103].

This approximate marginal likelihood can be interpreted as the marginal likelihood of a specific GP defined on the original data set $(\mathbf{X}, \mathbf{y})$. In this GP, the original kernel k is replaced by the kernel $\mathrm{k}_{\mathrm{FITC}}$. With $\mathbb{I}$ denoting the indicator function, it is defined as

$$\begin{aligned} \mathcal{Q}(\mathbf{a}, \mathbf{b}) &\coloneqq \mathbf{K}_{\mathbf{au}} \mathbf{K}_{\mathbf{uu}}^{-1} \mathbf{K}_{\mathbf{ub}} \\ \mathrm{k}_{\mathrm{FITC}}(\mathbf{a}, \mathbf{b}) &\coloneqq \mathcal{Q}(\mathbf{a}, \mathbf{b}) + \mathbb{I}(\mathbf{a} = \mathbf{b})\, (\mathrm{k}(\mathbf{a}, \mathbf{b}) - \mathcal{Q}(\mathbf{a}, \mathbf{b}))\,. \end{aligned} \tag{2.55}$$

This kernel is equal to k when both arguments are identical and equal to $\mathcal{Q}$ everywhere else. For well-chosen inducing inputs, $\mathbf{Q}_{\mathbf{ff}}$ is a low-rank approximation of $\mathbf{K}_{\mathbf{ff}}$ [103].

The inducing inputs $\mathbf{Z}$ are hidden in the kernel matrix $\mathbf{K_{uu}}$ and are additional hyper-parameters to this kernel. The predictive posterior $\mathbf{f}_*$ of the test points $\mathbf{X}_*$ is then given by

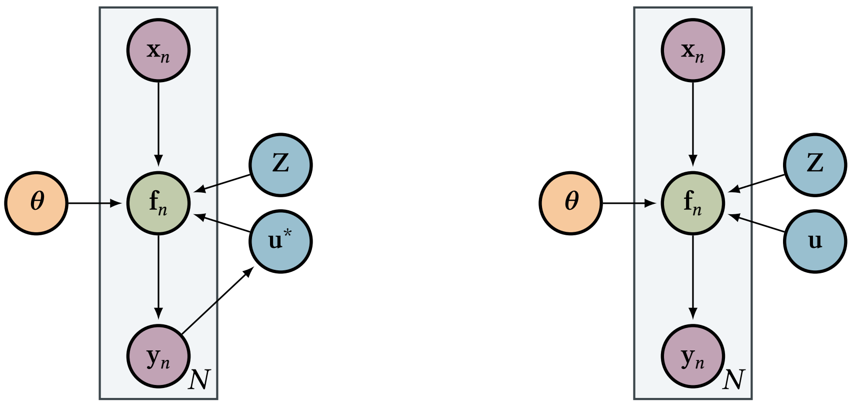
$$\begin{aligned}\mathrm{p}(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y}, \mathbf{Z}) = \mathcal{N}(\mathbf{f}_* \mid \boldsymbol{\mu}_*, \Sigma_*), \text{ where}&\\ \boldsymbol{\mu}_* &= \mathbf{Q_{*f}}\left(\mathbf{Q_{ff}} + \operatorname{diag}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) + \sigma^2 \mathrm{I}\right)^{-1} \mathbf{y} \\ \Sigma_* &= \mathbf{K}_{**} - \mathbf{Q_{*f}}\left(\mathbf{Q_{ff}} + \operatorname{diag}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) + \sigma^2 \mathrm{I}\right)^{-1} \mathbf{Q_{f*}}.\end{aligned} \tag{2.56}$$

and $\mathbf{Q_{ff}} \coloneqq \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uf}}$ obtained by inserting the kernel definition into Lemma 13. This formulation of the predictive posterior for the FITC approximation still requires the inversion of matrices of size $N \times N$ and therefore does not offer computational improvements. Using the matrix inversion lemma [84], they can be rewritten in the form

$$\begin{aligned}\boldsymbol{\mu}_* &= \mathbf{K_{*u}}\mathbf{B}^{-1}\mathbf{K_{uf}}\left(\operatorname{diag}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) + \sigma^2 \mathrm{I}\right)^{-1} \mathbf{y} \\ \Sigma_* &= \mathbf{K}_{**} - \mathbf{K_{*u}}\left(\mathbf{K_{uu}^{-1}} - \mathbf{B}^{-1}\right)\mathbf{K_{u*}} \\ \mathbf{B} &= \mathbf{K_{uu}} + \mathbf{K_{uf}}\left(\operatorname{diag}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) + \sigma^2 \mathrm{I}\right)^{-1} \mathbf{K_{fu}},\end{aligned} \tag{2.57}$$

which only involves the inversion of $M \times M$ matrices and one diagonal $N \times N$ matrix. Implemented this way, the calculation of all terms independent of the test points has a complexity of $\mathcal{O}(NM^2)$ and predicting individual means and variances takes $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ time respectively. The space requirement also drops to $\mathcal{O}(M^2)$. Since the positions of the inducing inputs $\mathbf{Z}$ are additional hyper-parameters in $\mathrm{k}_{\mathrm{FITC}}$, they can be chosen together with the hyper-parameters of the original kernel $\boldsymbol{\theta}$ using maximum likelihood. This optimization chooses the positions in such a way that together with appropriate other hyper-parameters, the original data is represented as well as possible. Figure 2.8 shows that a surprisingly small number of inducing inputs can be enough to represent the dynamics of a function.

However, with a large number of inducing inputs, the number of hyper-parameters can grow large as well. A large number of hyperparameters implies a danger of overfitting since the altered Gaussian process has no direct connection to the original Gaussian process over the complete training set. It is also not clear what properties a set of training inputs $\mathbf{Z}$ must fulfill such that it recovers an original GP well. To address these issues, a variational formulation of sparse GPs using inducing observations formulated originally in [56, 113] is discussed next.



(a) SGPR **(b)** SVGP

Figure 2.9: Graphical models of variational sparse GP approximations with inducing inputs. The latent function values $\mathbf{f}_n$ are assumed to be independent given the variational parameters $\mathbf{Z}$, $\mathbf{u}$ and hyper-parameters $\boldsymbol{\theta}$. Since the observations $\mathbf{y}_n$ inform the optimal $q^*(\mathbf{u})$ distribution in the SGPR approximations, they are not conditionally independent in the SGPR approximation. In the SVGP approximation, the parameters for $q(\mathbf{u})$ are part of the variational bound, which implies conditional independence between observations and enables stochastic optimization techniques.

2.6 Variational Sparse Gaussian Process Approximations

Titsias [113] introduced a variational interpretation of sparse GPs with inducing observations $(\mathbf{Z}, \mathbf{u})$. Similar to the FITC approximation discussed above, inducing observations are assumed to be generated from the latent function $\mathbf{u} = f(\mathbf{Z})$. Due to the consistency of GPs, the true data and inducing data are jointly Gaussian.

$$\mathrm{p}(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\begin{pmatrix} \mathbf{f} \\ \mathbf{u} \end{pmatrix} \middle| \mathbf{0}, \begin{pmatrix} \mathbf{K_{ff}} & \mathbf{K_{fu}} \\ \mathbf{K_{uf}} & \mathbf{K_{uu}} \end{pmatrix}\right) \tag{2.58}$$

Instead of defining a new GP on the inducing data, we want to choose the M inducing locations such that the original GP defined on the N true data points is approximated as closely as possible.

More formally, we consider the predictive posterior of the augmented GP containing both the true and inducing data given by

$$\mathrm{p}(\mathbf{f}^* \,|\, \mathbf{y}) = \int \mathrm{p}(\mathbf{f}^* \,|\, \mathbf{f}, \mathbf{u})\, \mathrm{p}(\mathbf{f}, \mathbf{u} \,|\, \mathbf{y})\, \mathrm{d}\mathbf{f}\, \mathrm{d}\mathbf{u}, \tag{2.59}$$

where we drop the conditioning on $\mathbf{X}$ and $\mathbf{Z}$ for notational simplicity. We adopt this convention for the rest of this chapter. The inducing observations are optimal if $\mathbf{f}^*$ and $\mathbf{f}$ are independent given $\mathbf{u}$ in

$$\begin{aligned} p(\mathbf{f}^* \mid \mathbf{f}, \mathbf{u}) &= p(\mathbf{f}^* \mid \mathbf{u}) \\ p(\mathbf{f}, \mathbf{u} \mid \mathbf{y}) &= p(\mathbf{f} \mid \mathbf{u})\, p(\mathbf{u} \mid \mathbf{y}). \end{aligned} \tag{2.60}$$

In this case, $\mathbf{u}$ is said to be a sufficient statistic for $\mathbf{f}$, capturing all information contained in the latter. In practice, it is hard to find $(\mathbf{Z}, \mathbf{u})$ that are indeed a sufficient statistic for $\mathbf{f}$. We will approximate this situation with a variational distribution $q(\mathbf{f}, \mathbf{u})$, thereby formulating a variational approximation to the original GP. Due to the joint Gaussian distribution in (2.58), it is convenient to consider the factorization

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u})\, q(\mathbf{u}), \tag{2.61}$$

where $p(\mathbf{f} \mid \mathbf{u})$ is a standard Gaussian conditional. Assuming $\mathbf{u}$ is indeed a sufficient statistic for $\mathbf{f}$, the variational predictive posterior reduces to

$$\begin{aligned} q(\mathbf{f}^*) &= \int p(\mathbf{f}^* \mid \mathbf{f}, \mathbf{u})\, q(\mathbf{f}, \mathbf{u}) \,\mathrm{d}\mathbf{f} \,\mathrm{d}\mathbf{u} \\ &= \int p(\mathbf{f}^* \mid \mathbf{u})\, p(\mathbf{f} \mid \mathbf{u})\, q(\mathbf{u}) \,\mathrm{d}\mathbf{f} \,\mathrm{d}\mathbf{u} \\ &= \int p(\mathbf{f}^* \mid \mathbf{u})\, q(\mathbf{u}) \,\mathrm{d}\mathbf{u}. \end{aligned} \tag{2.62}$$

To formulate a variational lower bound on the original marginal likelihood $\mathscr{L}^{\text{GP}}$ in (2.41), we have to decide how to choose $\mathbf{Z}$ and $q(\mathbf{u})$ such that as much information of $\mathbf{f}$ is captured as possible. We will discuss two approaches. First, we derive the optimal choice of $q(\mathbf{u})$ given a set of inducing inputs $\mathbf{Z}$. Because calculating this optimal choice is computationally expensive, we will then show how to improve performance through further approximation.

Optimal Inducing Outputs

We assume a free-form variational distribution $q(\mathbf{u})$ to derive the variational lower bound to the likelihood $\mathscr{L}^{\text{SGPR}}$ of the augmented model.

$$\begin{aligned} \mathscr{L}^{\text{SGPR}}(\boldsymbol{\theta}, \mathbf{Z}, q(\mathbf{u})) &= -\log p(\mathbf{y} \mid \boldsymbol{\theta}, \mathbf{Z}, q(\mathbf{u})) \\ &= -\log \int p(\mathbf{y} \mid \mathbf{f})\, p(\mathbf{f} \mid \mathbf{u})\, p(\mathbf{u}) \,\mathrm{d}\mathbf{f} \,\mathrm{d}\mathbf{u} \\ &= -\log \int q(\mathbf{f}, \mathbf{u}) \frac{p(\mathbf{y} \mid \mathbf{f})\, p(\mathbf{f} \mid \mathbf{u})\, p(\mathbf{u})}{q(\mathbf{f}, \mathbf{u})} \,\mathrm{d}\mathbf{f} \,\mathrm{d}\mathbf{u}. \end{aligned} \tag{2.63}$$

We bound the likelihood using Jensen's inequality [13] which states that for convex functions f and integrable functions g it holds that

$$f\left(\int g(x)\,\mathrm{d}x\right) \leq \int f(g(x))\,\mathrm{d}x. \tag{2.64}$$

Since the natural logarithm is concave we have

$$\begin{aligned}
\mathcal{L}^{\mathrm{SGPR}}(\boldsymbol{\theta}, \mathbf{Z}, \mathrm{q}(\mathbf{u})) &\geq -\int \mathrm{q}(\mathbf{f}, \mathbf{u}) \log \frac{\mathrm{p}(\mathbf{y} \,|\, \mathbf{f})\, \mathrm{p}(\mathbf{f} \,|\, \mathbf{u})\, \mathrm{p}(\mathbf{u})}{\mathrm{q}(\mathbf{f}, \mathbf{u})} \,\mathrm{d}\mathbf{f}\,\mathrm{d}\mathbf{u} \\
&= -\int \mathrm{p}(\mathbf{f} \,|\, \mathbf{u})\, \mathrm{q}(\mathbf{u}) \log \frac{\mathrm{p}(\mathbf{y} \,|\, \mathbf{f})\, \mathrm{p}(\mathbf{f} \,|\, \mathbf{u})\, \mathrm{p}(\mathbf{u})}{\mathrm{p}(\mathbf{f} \,|\, \mathbf{u})\, \mathrm{q}(\mathbf{u})} \,\mathrm{d}\mathbf{f}\,\mathrm{d}\mathbf{u} \\
&= -\int \mathrm{p}(\mathbf{f} \,|\, \mathbf{u})\, \mathrm{q}(\mathbf{u}) \log \frac{\mathrm{p}(\mathbf{y} \,|\, \mathbf{f})\, \mathrm{p}(\mathbf{u})}{\mathrm{q}(\mathbf{u})} \,\mathrm{d}\mathbf{f}\,\mathrm{d}\mathbf{u} \\
&= -\int \mathrm{q}(\mathbf{u}) \left(\int \mathrm{p}(\mathbf{f} \,|\, \mathbf{u}) \log \mathrm{p}(\mathbf{y} \,|\, \mathbf{f}) \,\mathrm{d}\mathbf{f} + \log \frac{\mathrm{p}(\mathbf{u})}{\mathrm{q}(\mathbf{u})} \right) \mathrm{d}\mathbf{u} \\
&= -\int \mathrm{q}(\mathbf{u}) \left(\mathbb{E}_{\mathrm{p}(\mathbf{f}|\mathbf{u})}[\log \mathrm{p}(\mathbf{y} \,|\, \mathbf{f})] + \log \frac{\mathrm{p}(\mathbf{u})}{\mathrm{q}(\mathbf{u})} \right) \mathrm{d}\mathbf{u} \\
&= \mathbb{E}_{\mathrm{q}(\mathbf{f})}[\log \mathrm{p}(\mathbf{y} \,|\, \mathbf{f})] - \mathrm{KL}(\mathrm{q}(\mathbf{u}) \,\|\, \mathrm{p}(\mathbf{u})),
\end{aligned} \tag{2.65}$$

dropping the conditioning on $\boldsymbol{\theta}$ and $\mathbf{Z}$. We assume a Gaussian likelihood $\mathrm{p}(\mathbf{y} \,|\, \mathbf{f}) = \mathcal{N}(\mathbf{y} \,|\, \mathbf{f}, \sigma_n^2 \mathrm{I})$. The expectation of the log-likelihood can be evaluated analytically [84] and is given by

$$\begin{aligned}
\mathbb{E}_{\mathrm{p}(\mathbf{f}|\mathbf{u})}[\log \mathrm{p}(\mathbf{y} \,|\, \mathbf{f})] &= \int \mathrm{p}(\mathbf{f} \,|\, \mathbf{u}) \log \mathrm{p}(\mathbf{y} \,|\, \mathbf{f}) \,\mathrm{d}\mathbf{f} \\
&= \int \log \mathcal{N}\big(\mathbf{y} \,|\, \mathbf{f}, \sigma_n^2 \mathrm{I}\big)\, \mathcal{N}\big(\mathbf{f} \,|\, \mathbf{K}_{\mathbf{fu}} \mathbf{K}_{\mathbf{uu}}^{-1} \mathbf{u}, \mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}}\big) \,\mathrm{d}\mathbf{f} \\
&= \log \mathcal{N}\big(\mathbf{y} \,|\, \mathbf{K}_{\mathbf{fu}} \mathbf{K}_{\mathbf{uu}}^{-1} \mathbf{u}, \sigma_n^2 \mathrm{I}\big) - \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}}) \\
&= \log G(\mathbf{u}) - \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}}),
\end{aligned} \tag{2.66}$$

where we set $G(\mathbf{u}) = \mathcal{N}\big(\mathrm{y}\,|\,\mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}, \sigma_n^2\mathrm{I}\big)$, $\mathbf{Q_{ff}} = \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{uf}}$ and where $\mathrm{tr}(\mathbf{M})$ denotes the trace of the matrix $\mathbf{M}$. Inserting (2.66) into (2.65) yields

$$\begin{aligned}
\mathscr{L}^{\mathrm{SGPR}}(\boldsymbol{\theta}, \mathbf{Z}, \mathrm{q}(\mathbf{u})) &\geq -\int \mathrm{q}(\mathbf{u}) \left(\mathbb{E}_{\mathrm{p}(\mathbf{f}|\mathbf{u})}[\log \mathrm{p}(\mathrm{y}\,|\,\mathbf{f})] + \log \frac{\mathrm{p}(\mathbf{u})}{\mathrm{q}(\mathbf{u})} \right) \mathrm{d}\mathbf{u} \\
&= -\int \mathrm{q}(\mathbf{u}) \left(\log G(\mathbf{u}) - \frac{1}{2\sigma_n^2} \mathrm{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) + \log \frac{\mathrm{p}(\mathbf{u})}{\mathrm{q}(\mathbf{u})} \right) \mathrm{d}\mathbf{u} \\
&= -\int \mathrm{q}(\mathbf{u}) \log \frac{G(\mathbf{u})\,\mathrm{p}(\mathbf{u})}{\mathrm{q}(\mathbf{u})} \,\mathrm{d}\mathbf{u} + \frac{1}{2\sigma_n^2} \mathrm{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}).
\end{aligned} \tag{2.67}$$

The distribution $\mathrm{q}(\mathbf{u})$ is chosen optimally if it maximizes the variational bound $\mathscr{L}^{\mathrm{SGPR}}$ in

$$\begin{aligned}
&\underset{\mathrm{q}(\mathbf{u})}{\arg\max}\, \mathscr{L}^{\mathrm{SGPR}}(\boldsymbol{\theta}, \mathbf{Z}, \mathrm{q}(\mathbf{u})) \\
&\quad = \underset{\mathrm{q}(\mathbf{u})}{\arg\max} -\int \mathrm{q}(\mathbf{u}) \log \frac{G(\mathbf{u})\,\mathrm{p}(\mathbf{u})}{\mathrm{q}(\mathbf{u})} \,\mathrm{d}\mathbf{u} + \frac{1}{2\sigma_n^2} \mathrm{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) \\
&\quad = \underset{\mathrm{q}(\mathbf{u})}{\arg\max} -\int \mathrm{q}(\mathbf{u}) \log \frac{G(\mathbf{u})\,\mathrm{p}(\mathbf{u})}{\mathrm{q}(\mathbf{u})} \,\mathrm{d}\mathbf{u} \\
&\quad = \underset{\mathrm{q}(\mathbf{u})}{\arg\min}\, \mathrm{KL}(\mathrm{q}(\mathbf{u}) \,\|\, G(\mathbf{u})\,\mathrm{p}(\mathbf{u})),
\end{aligned} \tag{2.68}$$

where $\mathrm{KL}(\mathrm{q} \,\|\, \mathrm{p})$ denotes the Kullback-Leibler divergence. This divergence is minimized if q and p are proportional and $G(\mathbf{u})\,\mathrm{p}(\mathbf{u})$ is a product of Gaussians

$$\mathrm{q}^*(\mathbf{u}) \propto G(\mathbf{u})\,\mathrm{p}(\mathbf{u}) = \mathcal{N}\big(\mathrm{y}\,|\,\mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{u}, \sigma_n^2\mathrm{I}\big)\, \mathcal{N}(\mathbf{u}\,|\,\mathbf{0}, \mathbf{K_{uu}}), \tag{2.69}$$

where $\mathrm{p}(\mathbf{u})$ is due to the GP prior in (2.58). Since the product of two Gaussian densities is an un-normalized Gaussian density, the optimal choice $\mathrm{q}^*(\mathbf{u})$ is a Gaussian given by

$$\begin{aligned}
\mathrm{q}^*(\mathbf{u}) &= \frac{G(\mathbf{u})\,\mathrm{p}(\mathbf{u})}{\int G(\mathbf{u})\,\mathrm{p}(\mathbf{u})\,\mathrm{d}\mathbf{u}} = \mathcal{N}(\mathbf{u}\,|\,\boldsymbol{\mu}_u, \Sigma_u), \text{ with} \\
\boldsymbol{\mu}_u &= \sigma_n^{-2}\mathbf{K_{uu}}\mathbf{B}^{-1}\mathbf{K_{uf}}\mathrm{y} \\
\Sigma_u &= \mathbf{K_{uu}}\mathbf{B}^{-1}\mathbf{K_{uu}} \\
\mathbf{B} &= \mathbf{K_{uu}} + \sigma_n^{-2}\mathbf{K_{uf}}\mathbf{K_{fu}}.
\end{aligned} \tag{2.70}$$

Inserting q^* into (2.67) yields the final bound

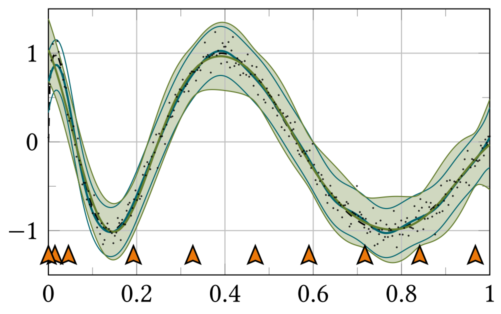
$$\begin{aligned}
\mathscr{L}^{\text{SGPR}}(\boldsymbol{\theta}, \mathbf{Z}) &\geq \mathscr{L}^{\text{SGPR}}(\boldsymbol{\theta}, \mathbf{Z}, \text{q}^*(\mathbf{u})) \\
&= -\int \text{q}^*(\mathbf{u}) \log \frac{G(\mathbf{u})\,\text{p}(\mathbf{u})}{\text{q}^*(\mathbf{u})} \,\text{d}\mathbf{u} + \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) \\
&= -\int \frac{G(\mathbf{u})\,\text{p}(\mathbf{u})}{\int G(\mathbf{u})\,\text{p}(\mathbf{u})\,\text{d}\mathbf{u}} \log \frac{G(\mathbf{u})\,\text{p}(\mathbf{u})}{\frac{G(\mathbf{u})\,\text{p}(\mathbf{u})}{\int G(\mathbf{u})\,\text{p}(\mathbf{u})\,\text{d}\mathbf{u}}} \,\text{d}\mathbf{u} + \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) \\
&= -\log \int G(\mathbf{u})\,\text{p}(\mathbf{u})\,\text{d}\mathbf{u} + \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) \\
&= -\log \mathcal{N}\left(\mathbf{y} \,\middle|\, \mathbf{0}, \mathbf{Q_{ff}} + \sigma_n^2 \text{I}\right) + \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}),
\end{aligned} \tag{2.71}$$

which can be maximized to jointly find hyper-parameters $\boldsymbol{\theta}$ and the variational parameters $\mathbf{Z}$. Evaluating this bound takes $\mathcal{O}(NM^2)$ time. In contrast to the FITC approximation, where inducing inputs are additional hyper-parameters to the model that can lead to overfitting, adding inducing inputs $\mathbf{Z}$ to the variational model only improves the approximation to the full GP.

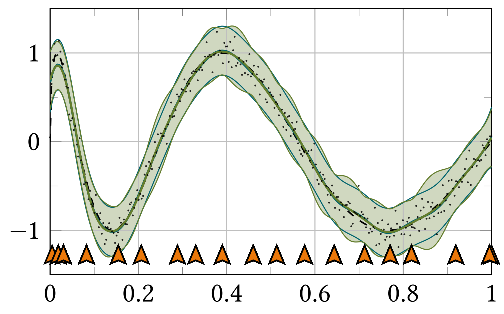
Figure 2.9a shows the graphical model for the variational GP approximation using the optimal $\text{q}^*(\mathbf{u})$ distribution. The inducing inputs Ⓩ are referred to as variational parameters. While the SGPR model is more computationally efficient, the dependencies between different observations are not wholely removed. The optimal choice for $\text{q}^*(\mathbf{u})$ depends on the observations in (2.70). Because of this, the bound does not factorize along the data, preventing stochastic variational inference techniques [56] and requiring the bound to be evaluated jointly for all data points. Additionally, evaluating the predictive posterior (2.62) takes $\mathcal{O}(M^3 + MN)$ time. For large N, linear growth with the amount of observations can be prohibitive. We will now consider the SVGP model introduced by Hensman et al. [56] that avoids this growth.

Approximate Inducing Outputs

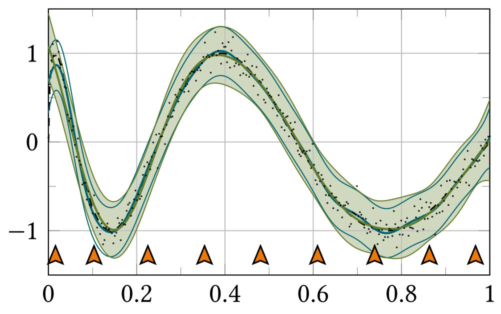
In the SGPR approximation, a linear dependency on the number of points in the true data set is introduced through $\text{q}^*(\mathbf{u})$. We have shown in (2.70) that this optimal choice is a Gaussian. Instead of using $\text{q}^*(\mathbf{u})$, the SVGP model shown in Figure 2.9b uses a free-form Gaussian $\text{q}(\mathbf{u}) = \mathcal{N}(\mathbf{u} \,|\, \mathbf{m}, \text{S})$. The $\mathcal{O}(M^2)$ variational parameters in $\mathbf{m}$ and S are optimized jointly with $\mathbf{Z}$ in a variational bound. Poor choices of $\mathbf{m}$ and S can only



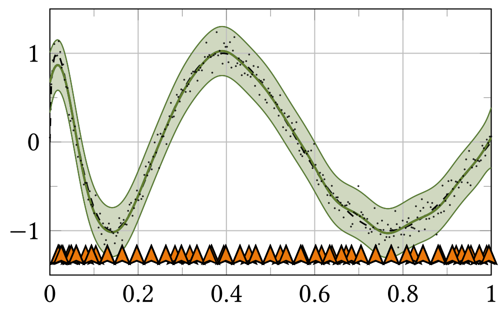
(a) FITC with $M = 10$



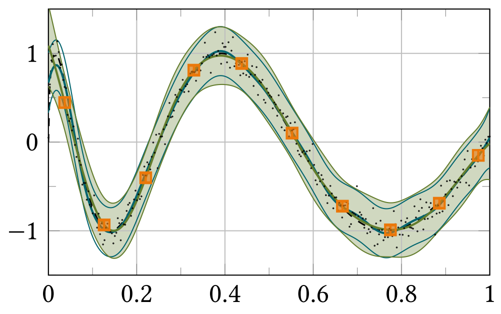
(b) FITC with $M = 300$



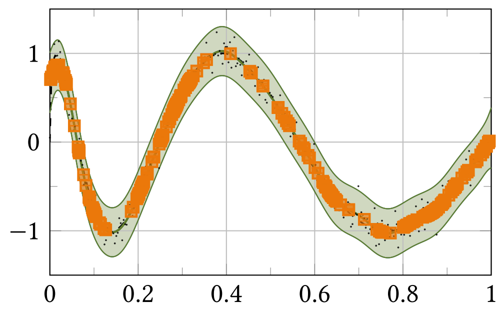
(c) SGPR with $M = 10$



(d) SGPR with $M = 300$



(e) SVGP with $M = 10$



(f) SVGP with $M = 300$

Figure 2.10: A comparison of sparse GP approximations with inducing inputs. While the FITC approximation does not converge to the full GP for large M, both variational approaches do. For SGPR and FITC, the inducing outputs are marginalized and only the x-coordinates are meaningful. For SVGP, inducing outputs are optimized as well.

worsen the variational approximation compared to the optimal choice in $\mathscr{L}^{\mathrm{SGPR}}$ but does not alter the model.

To derive the variational bound $\mathscr{L}^{\mathrm{SVGP}}$, we start with the bound in (2.67). Instead of inserting the optimal $\mathrm{q}^*(\mathbf{u})$, we reformulate the bound to recover an expectation and KL-divergence in

$$\begin{aligned}
&\mathscr{L}^{\mathrm{SVGP}}(\boldsymbol{\theta}, \mathbf{Z}, \mathbf{m}, \mathrm{S}) \\
&\quad \geq - \int \mathrm{q}(\mathbf{u}) \log \frac{G(\mathbf{u})\, \mathrm{p}(\mathbf{u})}{\mathrm{q}(\mathbf{u})} \,\mathrm{d}\mathbf{u} + \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) \\
&\quad = - \int \mathrm{q}(\mathbf{u}) \log \frac{G(\mathbf{u})\, \mathrm{p}(\mathbf{u})}{\mathrm{q}(\mathbf{u})} \,\mathrm{d}\mathbf{u} + \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) \\
&\quad = - \int \log G(\mathbf{u})\, \mathrm{q}(\mathbf{u}) \,\mathrm{d}\mathbf{u} + \mathrm{KL}(\mathrm{q}(\mathbf{u}) \,\|\, \mathrm{p}(\mathbf{u})) + \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) \\
&\quad = - \mathbb{E}_{\mathrm{q}(\mathbf{u})}[\log G(\mathbf{u})] + \mathrm{KL}(\mathrm{q}(\mathbf{u}) \,\|\, \mathrm{p}(\mathbf{u})) + \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}).
\end{aligned} \tag{2.72}$$

Similar to (2.66), this expectation can be evaluated analytically and is given by

$$\begin{aligned}
\mathbb{E}_{\mathrm{q}(\mathbf{u})}[\log G(\mathbf{u})] &= \int \log G(\mathbf{u})\, \mathrm{q}(\mathbf{u}) \,\mathrm{d}\mathbf{u} \\
&= \int \log \mathcal{N}\big(\mathbf{y} \,\big|\, \mathbf{K_{fu}} \mathbf{K_{uu}^{-1}} \mathbf{u}, \sigma_n^2 \mathrm{I}\big)\, \mathcal{N}(\mathbf{u} \,|\, \mathbf{m}, \mathrm{S}) \,\mathrm{d}\mathbf{u} \\
&= \log \mathcal{N}\big(\mathbf{y} \,\big|\, \mathbf{K_{fu}} \mathbf{K_{uu}^{-1}} \mathbf{m}, \sigma_n^2 \mathrm{I}\big) - \frac{1}{2\sigma_n^2} \operatorname{tr}\big(\mathbf{K_{fu}} \mathbf{K_{uu}^{-1}} \mathrm{S} \mathbf{K_{uu}^{-1}} \mathbf{K_{uf}}\big).
\end{aligned} \tag{2.73}$$

Inserting (2.73) into (2.72) yields the final bound for the SVGP model given by

$$\begin{aligned}
&\mathscr{L}^{\mathrm{SVGP}}(\boldsymbol{\theta}, \mathbf{Z}, \mathbf{m}, \mathrm{S}) \\
&\quad \geq - \mathbb{E}_{\mathrm{q}(\mathbf{u})}[\log G(\mathbf{u})] + \mathrm{KL}(\mathrm{q}(\mathbf{u}) \,\|\, \mathrm{p}(\mathbf{u})) + \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) \\
&\quad = - \log \mathcal{N}\big(\mathbf{y} \,\big|\, \mathbf{K_{fu}} \mathbf{K_{uu}^{-1}} \mathbf{m}, \sigma_n^2 \mathrm{I}\big) + \mathrm{KL}(\mathrm{q}(\mathbf{u}) \,\|\, \mathrm{p}(\mathbf{u})) \\
&\qquad + \frac{1}{2\sigma_n^2} \operatorname{tr}(\mathbf{K_{ff}} - \mathbf{Q_{ff}}) + \frac{1}{2\sigma_n^2} \operatorname{tr}\big(\mathbf{K_{fu}} \mathbf{K_{uu}^{-1}} \mathrm{S} \mathbf{K_{uu}^{-1}} \mathbf{K_{uf}}\big).
\end{aligned} \tag{2.74}$$

The KL-divergence is a divergence of two Gaussians which can be evaluated analytically. While evaluating the complete bound still takes $\mathcal{O}(NM^2)$, the data-likelihood term is given by a diagonal Gaussian which factorizes along the data. This enables stochastic optimization techniques like mini-batching, greatly increasing the scalability of the variational GP approximation to large data sets [56]. Similarly, evaluating the predictive posterior (2.62) now takes $\mathcal{O}(M^3)$ time, removing the dependency on the training data completely. Figure 2.10 compares predictive posteriors for the FITC, SGPR and SVGP approximations. While the approximations show similar results with small amounts of inducing inputs, both SGPR and SVGP converge to the original GP for large M while the FITC approximation does not. Because the SGPR approximation does not explicitly represent inducing outputs, only the input locations are meaningful, similar to the FITC approximation. SVGP directly infers a posterior about latent function values, which are shown in the figure.

The $\mathcal{O}(M^3)$ computational cost due to the inversion of $\mathbf{K_{uu}}$ can still be prohibitive for models with a large number of inducing points. To reduce the number of required points, recent work explored how Bayesian inference can be employed in the search for good inducing point locations [58, 90]. There have also been multiple extensions to the variational inducing point approach to reduce the computational cost further. One approach is to impose a grid structure on the inducing inputs $\mathbf{Z}$ [122]. Instead of optimizing their location, the position of a large number of inputs is fixed to perform fast computations exploiting the structure. While this approach suffers from the curse of dimensionality, it can increase performance for low input dimensionalities. Another approach is to orthogonally decouple the computations for predictive means and variances [21, 93, 98]. This decoupling allows the calculation of the predictive mean in linear time, allowing for a larger number of inducing points for the mean.

2.7 Hierarchical Gaussian Processes

While GPs offer a principled non-parametric approach to representing distributions over functions, they can be restrictive in the functions they can represent. Many extensions to standard GPs have been studied which introduce additional hierarchical structure such as GP latent variable models [33, 114], warped GP models [74, 102] or mixtures of experts models [75, 86, 115]. The hierarchical structure typically introduces the requirement to propagate uncertainties through Gaussian processes, which is analytically intractable in most cases. Many extensions can be interpreted as special cases of the hierarchical or deep Gaussian process model [31, 32, 73] in which a nested generative process is formulated by using the output of one GP as the input of another.

In a deep GP, rather than assuming that observational data is generated through a draw from a single GP which is then corrupted by noise in $\mathbf{y} = f(\mathbf{x}) + \epsilon$, we assume

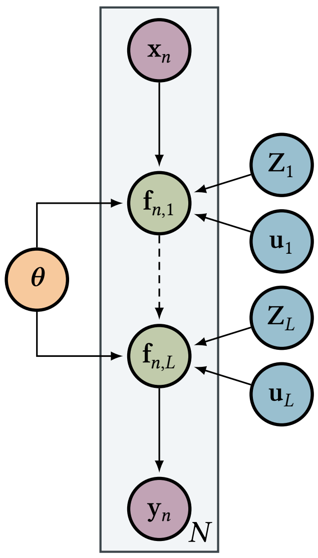


Figure 2.11: Graphical model of a variational deep GP approximation with inducing inputs. A deep GP is a composition of L functions where each f_l is drawn from a GP. Similar to shallow variational GP approximations, each GP is augmented with inducing observations.

that data is generated via a composition of L functions

$$\mathbf{y} = f_L(f_{L-1}(\cdots(f_1(\mathbf{X})))) + \epsilon, \tag{2.75}$$

where each f_l is drawn from a GP. This new compositional prior $f_L \circ \cdots \circ f_1$ is no longer a Gaussian process [43] and can represent a broader set of functions.

Figure 2.11 shows the graphical model of a deep GP. We represent the result of applying the first l functions $f_l \circ \cdots \circ f_1$ on the inputs $\mathbf{X}$ as $\mathbf{f}_l$ and identify $\mathbf{f}_0 = \mathbf{X}$. The joint probability of the function values and the outputs can then be written as

$$\begin{aligned} \mathrm{p}(\mathbf{y}, \mathbf{f}_1, \dots, \mathbf{f}_L \,|\, \mathbf{X}) = \mathrm{p}(\mathbf{y} \,|\, \mathbf{f}_L) \prod_{l=1}^{L} \mathrm{p}(\mathbf{f}_l \,|\, \mathbf{f}_{l-1}), \text{ with} \\ \mathbf{y} \mid \mathbf{f}_L \sim \mathcal{N}(\mathbf{f}_L, \sigma_n^2 \mathrm{I}) \\ \mathbf{f}_l \mid \mathbf{f}_{l-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{f}_l \mathbf{f}_l} + \sigma_l^2 \mathrm{I}). \end{aligned} \tag{2.76}$$

The structure of the conditionals $\mathrm{p}(\mathbf{f}_l \,|\, \mathbf{f}_{l-1})$ is implied by the assumption that the f_l are drawn from GPs and (2.41). Note that following the original formulation in [32], we assume independent Gaussian noise σ_l^2 as part of every function f_l. We will denote the kernel matrix at the l^{th} level as $\mathbf{K}_{ll} = \mathbf{K}_{\mathbf{f}_l\mathbf{f}_l}$. The $\mathbf{f}_l$ are all latent variables, and inference over them is very challenging. Because they capture the result of applying multiple GPs, they are highly dependent on each other, thereby implying dependence between the different functions as well. We will now derive two extensions of the sparse variational approximations in Section 2.6 to the hierarchical case, nested variational compression [57] and doubly stochastic variational inference [94] to approximate the marginal likelihood of the deep Gaussian process model

$$\mathrm{p}(\mathbf{y} \,|\, \mathbf{X}) = \int \mathrm{p}(\mathbf{y}, \mathbf{f}_1, \dots, \mathbf{f}_L \,|\, \mathbf{X}) \,\mathrm{d}\mathbf{f}_1 \dots \mathrm{d}\mathbf{f}_L. \tag{2.77}$$

Nested Variational Compression

The challenge in deriving the marginal likelihood for deep GPs stems from the propagation of uncertainties through multiple GPs. In Section 2.6 and Figure 2.9b we showed how to achieve conditional independence between data points in a single GP by augmenting the model with a set of inducing observations. The SVGP model is a compressed representation in which a small number of inducing points represents the full GP defined on all observations. In nested variational compression (NVC) presented by Hensman et al. [57], this compression is applied recursively for all GPs in the deep GP hierarchy. To derive a tractable variational lower bound, an additional step is necessary to avoid the inter-layer cross-dependencies.

The bound in (2.73) can be used directly to approximate $\mathrm{p}(\mathbf{f}_1 \,|\, \mathbf{X})$ as the innermost function in a deep GP is a standard GP model. Our next goal is to derive a bound on the outputs of the second layer

$$\begin{aligned} \log \mathrm{p}(\mathbf{f}_2 \,|\, \mathbf{u}_2) &= \log \int \mathrm{p}(\mathbf{f}_2, \mathbf{f}_1, \mathbf{u}_1 \,|\, \mathbf{u}_2) \,\mathrm{d}\mathbf{f}_1 \,\mathrm{d}\mathbf{u}_1 \\ &= \log \int \mathrm{p}(\mathbf{f}_2, \,|\, \mathbf{u}_2, \mathbf{f}_1, \mathbf{u}_1) \,\mathrm{p}(\mathbf{f}_1, \mathbf{u}_1) \,\mathrm{d}\mathbf{f}_1 \,\mathrm{d}\mathbf{u}_1, \end{aligned} \tag{2.78}$$

that is, an expression in which the uncertainty about the $\mathbf{f}_1$ and the cross-layer dependencies on the $\mathbf{u}_1$ are both marginalized. We start by considering the relevant terms from (2.76) and apply (2.73) to marginalize $\mathbf{f}_1$ in

$$\begin{aligned}\log \mathrm{p}(\mathbf{f}_2 \mid \mathbf{u}_2, \mathbf{u}_1) &= \log \int \mathrm{p}(\mathbf{f}_2, \mathbf{f}_1 \mid \mathbf{u}_2, \mathbf{u}_1) \, \mathrm{d}\mathbf{f}_1 \\ &\geq \log \int \bar{\mathrm{p}}(\mathbf{f}_2 \mid \mathbf{u}_2, \mathbf{f}_1) \, \bar{\mathrm{p}}(\mathbf{f}_1 \mid \mathbf{u}_1) \\ &\qquad \cdot \exp\left(-\frac{1}{2\sigma_1^2} \operatorname{tr}(\mathbf{K}_{11} - \mathbf{Q}_{11}) - \frac{1}{2\sigma_2^2} \operatorname{tr}(\mathbf{K}_{22} - \mathbf{Q}_{22}) \right) \mathrm{d}\mathbf{f}_1 \\ &\geq \mathbb{E}_{\bar{\mathrm{p}}(\mathbf{f}_1 \mid \mathbf{u}_1)}[\log \bar{\mathrm{p}}(\mathbf{f}_2 \mid \mathbf{u}_2, \mathbf{f}_1)] \\ &\qquad - \mathbb{E}_{\bar{\mathrm{p}}(\mathbf{f}_1 \mid \mathbf{u}_1)}\left[\frac{1}{2\sigma_2^2} \operatorname{tr}(\mathbf{K}_{22} - \mathbf{Q}_{22}) \right] - \frac{1}{2\sigma_1^2} \operatorname{tr}(\mathbf{K}_{11} - \mathbf{Q}_{11}),\end{aligned} \tag{2.79}$$

where we write $\bar{\mathrm{p}}(\mathbf{f}_1 \mid \mathbf{u}_1) = \mathcal{N}(\mathbf{f}_1 \mid \boldsymbol{\mu}_1, \sigma_1^2 \mathrm{I})$ to incorporate the Gaussian noise in the latent space. Due to our assumption that $\mathbf{u}_1$ is a sufficient statistic for $\mathbf{f}_1$ we choose the Gaussians

$$\begin{aligned}\mathrm{q}(\mathbf{f}_1 \mid \mathbf{u}_1) &= \bar{\mathrm{p}}(\mathbf{f}_1 \mid \mathbf{u}_1), \text{ and} \\ \mathrm{q}(\mathbf{f}_1) &= \int \bar{\mathrm{p}}(\mathbf{f}_1 \mid \mathbf{u}_1) \, \mathrm{q}(\mathbf{u}_1) \, \mathrm{d}\mathbf{u}_1,\end{aligned} \tag{2.80}$$

and use another variational approximation to marginalize $\mathbf{u}_1$. This yields

$$\begin{aligned}\log \mathrm{p}(\mathbf{f}_2 \mid \mathbf{u}_2) &= \log \int \mathrm{p}(\mathbf{f}_2, \mathbf{u}_1 \mid \mathbf{u}_2) \, \mathrm{d}\mathbf{u}_1 \\ &= \log \int \mathrm{p}(\mathbf{f}_2 \mid \mathbf{u}_2, \mathbf{u}_1) \, \mathrm{p}(\mathbf{u}_1) \, \mathrm{d}\mathbf{u}_1 \\ &\geq \int \mathrm{q}(\mathbf{u}_1) \log \frac{\mathrm{p}(\mathbf{f}_2 \mid \mathbf{u}_2, \mathbf{u}_1) \, \mathrm{p}(\mathbf{u}_1)}{\mathrm{q}(\mathbf{u}_1)} \, \mathrm{d}\mathbf{u}_1 \\ &= \mathbb{E}_{\mathrm{q}(\mathbf{u}_1)}[\log \mathrm{p}(\mathbf{f}_2 \mid \mathbf{u}_1, \mathbf{u}_2)] - \mathrm{KL}(\mathrm{q}(\mathbf{u}_1) \,\|\, \mathrm{p}(\mathbf{u}_1)) \\ &\geq \mathbb{E}_{\mathrm{q}(\mathbf{u}_1)}\big[\mathbb{E}_{\bar{\mathrm{p}}(\mathbf{f}_1 \mid \mathbf{u}_1)}[\log \bar{\mathrm{p}}(\mathbf{f}_2 \mid \mathbf{u}_2, \mathbf{f}_1)]\big] - \mathrm{KL}(\mathrm{q}(\mathbf{u}_1) \,\|\, \mathrm{p}(\mathbf{u}_1)) \\ &\qquad - \frac{1}{2\sigma_1^2} \operatorname{tr}(\mathbf{K}_{11} - \mathbf{Q}_{11}) - \mathbb{E}_{\mathrm{q}(\mathbf{u}_1)}\left[\mathbb{E}_{\bar{\mathrm{p}}(\mathbf{f}_1 \mid \mathbf{u}_1)}\left[\frac{1}{2\sigma_2^2} \operatorname{tr}(\mathbf{K}_{22} - \mathbf{Q}_{22}) \right] \right] \\ &\geq \mathbb{E}_{\mathrm{q}(\mathbf{f}_1)}[\log \bar{\mathrm{p}}(\mathbf{f}_2 \mid \mathbf{u}_2, \mathbf{f}_1)], - \mathrm{KL}(\mathrm{q}(\mathbf{u}_1) \,\|\, \mathrm{p}(\mathbf{u}_1)) \\ &\qquad - \frac{1}{2\sigma_1^2} \operatorname{tr}(\mathbf{K}_{11} - \mathbf{Q}_{11}) - \frac{1}{2\sigma_2^2} \mathbb{E}_{\mathrm{q}(\mathbf{f}_1)}[\operatorname{tr}(\mathbf{K}_{22} - \mathbf{Q}_{22})],\end{aligned} \tag{2.81}$$

where we apply Fubini's theorem [46] to exchange the order of integration in the expected values. The expectations with respect to $\mathrm{q}(\mathbf{f}_1)$ involve expectations of kernel matrices, also called Ψ-statistics, in the same way as in [32] and are given by

$$\begin{aligned}\psi_2 &= \mathbb{E}_{q(\mathbf{f}_1)}\left[\operatorname{tr}\left(\mathbf{K}_{\mathbf{f}_2\mathbf{f}_2}\right)\right],\\ \mathbf{\Psi}_2 &= \mathbb{E}_{q(\mathbf{f}_1)}\left[\mathbf{K}_{\mathbf{f}_2\mathbf{u}_2}\right],\\ \mathbf{\Phi}_2 &= \mathbb{E}_{q(\mathbf{f}_1)}\left[\mathbf{K}_{\mathbf{u}_2\mathbf{f}_2}\mathbf{K}_{\mathbf{f}_2\mathbf{u}_2}\right].\end{aligned} \tag{2.82}$$

These Ψ-statistics can be computed analytically for multiple kernels, including the squared exponential kernel. To obtain the final formulation of the desired bound for $\log p(\mathbf{f}_2 \,|\, \mathbf{u}_2)$ we substitute (2.82) into (2.81) and get the analytically tractable bound

$$\begin{aligned}\log p(\mathbf{f}_2 \,|\, \mathbf{u}_2) \geq{} & \log \mathcal{N}\left(\mathbf{f}_2 \,\middle|\, \mathbf{\Psi}_2 \mathbf{K}_{\mathbf{u}_2\mathbf{u}_2}^{-1} \mathbf{m}_2, \sigma_2^2 \mathrm{I}\right) - \mathrm{KL}(q(\mathbf{u}_1) \,\|\, p(\mathbf{u}_1))\\ & - \frac{1}{2\sigma_1^2} \operatorname{tr}(\mathbf{K}_{11} - \mathbf{Q}_{11}) - \frac{1}{2\sigma_2^2}\left(\psi_2 - \operatorname{tr}\left(\mathbf{\Psi}_2 \mathbf{K}_{\mathbf{u}_2\mathbf{u}_2}^{-1}\right)\right)\\ & - \frac{1}{2\sigma_2^2} \operatorname{tr}\left(\left(\mathbf{\Phi}_2 - \mathbf{\Psi}_2^{\top}\mathbf{\Psi}_2\right) \mathbf{K}_{\mathbf{u}_2\mathbf{u}_2}^{-1} \left(\mathbf{m}_2 \mathbf{m}_2^{\top} + \mathrm{S}_2\right) \mathbf{K}_{\mathbf{u}_2\mathbf{u}_2}^{-1}\right).\end{aligned} \tag{2.83}$$

This bound is structurally similar to the SVGP bound in (2.73) but contains additional terms introduced by the propagation of uncertainties from $p(\mathbf{f}_1)$. To derive a variational bound for (2.77) where all $\mathbf{f}_l$ have been marginalized, we can apply the same steps as described above recursively, resulting in the nested variational compression lower bound given by

$$\begin{aligned}\mathcal{L}_{\mathrm{NVC}} \geq{} &\\ & \log \mathcal{N}\left(\mathbf{y} \,\middle|\, \mathbf{\Psi}_L \mathbf{K}_{\mathbf{u}_L\mathbf{u}_L}^{-1} \mathbf{m}_L, \sigma_n^2 \mathrm{I}\right) - \sum_{l=1}^{L} \mathrm{KL}(q(\mathbf{u}_l) \,\|\, p(\mathbf{u}_l))\\ & - \frac{1}{2\sigma_1^2} \operatorname{tr}(\mathbf{K}_{11} - \mathbf{Q}_{11}) - \sum_{l=2}^{L} \frac{1}{2\sigma_l^2}\left(\psi_l - \operatorname{tr}\left(\mathbf{\Psi}_l \mathbf{K}_{\mathbf{u}_l\mathbf{u}_l}^{-1}\right)\right)\\ & - \sum_{l=2}^{L} \frac{1}{2\sigma_l^2} \operatorname{tr}\left(\left(\mathbf{\Phi}_l - \mathbf{\Psi}_l^{\top}\mathbf{\Psi}_l\right) \mathbf{K}_{\mathbf{u}_l\mathbf{u}_l}^{-1} \left(\mathbf{m}_l \mathbf{m}_l^{\top} + \mathrm{S}_l\right) \mathbf{K}_{\mathbf{u}_l\mathbf{u}_l}^{-1}\right).\end{aligned} \tag{2.84}$$

Just as the SVGP bound, it factorizes along the data and enables stochastic optimization. However, depending on the kernel, the calculation of psi-statistics can be computationally expensive or analytically intractable, limiting the applicability of this bound to a limited set of hierarchical GP models.

Since the nested variational compression bound introduces a conditional independence assumption between layers given the approximations of the latent variables $q(\mathbf{f}_l)$, approximate predictions can be derived by recursively calculating $q(\mathbf{f}_1)$ to $q(\mathbf{f}_L)$, all of which are Gaussian. Given inputs $\mathbf{X}_*$, we recursively marginalize the intermediate layers

$$\begin{aligned} \mathrm{q}(\mathbf{f}_{l,*}) &= \int \mathrm{q}(\mathbf{f}_{l,*}, \mathbf{f}_{l-1,*}) \,\mathrm{d}\mathbf{f}_{l-1,*} \\ &= \int \mathrm{q}(\mathbf{f}_{l,*} \,|\, \mathbf{f}_{l-1,*}) \,\mathrm{q}(\mathbf{f}_{l-1,*}) \,\mathrm{d}\mathbf{f}_{l-1,*} \\ &= \mathbb{E}_{\mathrm{q}(\mathbf{f}_{l-1,*})}\left[\mathrm{q}(\mathbf{f}_{l,*} \,|\, \mathbf{f}_{l-1,*})\right] \\ &= \mathcal{N}\left(\mathbf{f}_{l,*} \,\middle|\, \bar{\boldsymbol{\mu}}_{l,*}, \bar{\Sigma}_{l,*}\right) \end{aligned} \tag{2.85}$$

with

$$\begin{aligned} \bar{\boldsymbol{\mu}}_{l,*} &= \boldsymbol{\Psi}_{l*} \mathbf{K}_{\mathbf{u}_l\mathbf{u}_l}^{-1} \mathbf{m}_l \\ \bar{\Sigma}_{\mathrm{l},*} &= \boldsymbol{\Psi}_{l*} \mathbf{K}_{\mathbf{u}_l\mathbf{u}_l}^{-1} \mathrm{S}_l \mathbf{K}_{\mathbf{u}_l\mathbf{u}_l}^{-1} \boldsymbol{\Psi}_{l*}^{\mathrm{T}}. \end{aligned}$$

For the first layer, the expectation collapses to usual SVGP predictions. The final function values are given by $\mathrm{q}(\mathbf{f}_L)$. While passing Gaussian variational messages through a deep GP to achieve conditional independence between layers is computationally convenient, it is a strong simplification as propagating a Gaussian distribution through a Gaussian process generally does not result in a Gaussian distribution. As discussed in more detail in [57], nested variational compression tends to underestimate uncertainties, especially when the functions f_l become more non-linear.

Doubly Stochastic Variational Inference

To overcome some of the limitations of nested variational compression, Salimbeni et al. [94] proposed the doubly stochastic variational inference (DSVI) approximation. Instead of relying on an explicit variational approximation for the $\mathbf{f}_l$, DSVI is based on the observation that due to the conditional independence assumption of the data introduced by the $\mathbf{u}_l$, function values $\mathbf{f}_{n,l}$ can be sampled independently and efficiently. Under the SVGP variational sufficient statistics assumptions, the posterior of a single GP can be written as

$$\begin{aligned} \mathrm{q}(\mathbf{f}_l \,|\, \mathbf{f}_{l-1}) &= \int \mathrm{q}(\mathbf{u}_l) \,\mathrm{p}(\mathbf{f}_l \,|\, \mathbf{u}_l, \mathbf{f}_{l-1}) \,\mathrm{d}\mathbf{u}_l \\ &= \int \mathrm{q}(\mathbf{u}_l) \prod_{n=1}^{N} \mathrm{p}\left(\mathbf{f}_{l,n} \,\middle|\, \mathbf{u}_l, \mathbf{f}_{l-1,n}\right) \mathrm{d}\mathbf{u}_l, \end{aligned} \tag{2.86}$$

which can be evaluated analytically, since it is a convolution of Gaussians. For the hierarchical case, we choose the same variational independence assumptions as in the NVC approximation above

$$\mathrm{q}(\mathbf{f}_1, \mathbf{u}_1, \dots, \mathbf{f}_L, \mathbf{u}_L) = \prod_{l=1}^{L} \mathrm{p}(\mathbf{f}_l \,|\, \mathbf{u}_l, \mathbf{f}_{l-1})\, \mathrm{q}(\mathbf{u}_l). \tag{2.87}$$

The factorization along the data still holds in the variational hierarchical case. We can formulate the marginal function value of the n^{th} data point by inserting (2.86) into (2.77) to obtain

$$\begin{aligned}\mathrm{q}\big(\mathbf{f}_{L,n}\big) &= \int \prod_{l=1}^{L-1} \mathrm{q}\big(\mathbf{f}_{l,n} \,\big|\, \mathbf{f}_{l-1,n}\big)\, \mathrm{d}\mathbf{f}_{l,n} \\ &= \int \prod_{l=1}^{L-1} \int \mathrm{q}(\mathbf{u}_l)\, \mathrm{p}\big(\mathbf{f}_{l,n} \,\big|\, \mathbf{u}_l, \mathbf{f}_{l-1,n}\big)\, \mathrm{d}\mathbf{u}_l\, \mathrm{d}\mathbf{f}_{l,n}.\end{aligned} \tag{2.88}$$

A consequence of this formulation is that drawing a sample from $\mathrm{q}\big(\mathbf{f}_{L,n}\big)$ can be achieved via ancestral sampling through the hierarchical GP model by drawing samples from the different $\mathrm{q}\big(\mathbf{f}_{l,n}\big)$ in turn. Instead of passing Gaussian messages as in the NVC approximation, we can draw Monte-Carlo samples from $\mathrm{q}\big(\mathbf{f}_{L,n}\big)$ directly. Because the data are independent under the variational assumptions, we can draw independent univariate samples per observation. Sampling $\mathrm{q}\big(\mathbf{f}_{L,n}\big)$ directly also allows us to sample the bound in (2.65) that generalizes to the hierarchical case and is given by

$$\mathscr{L}_{\text{DSVI}} \geq \sum_{n=1}^{N} \mathbb{E}_{\mathrm{q}(\mathbf{f}_{L,n})}\big[\log \mathrm{p}\big(\mathbf{y}_n \,\big|\, \mathbf{f}_{L,n}\big)\big] - \sum_{l=1}^{L} \mathrm{KL}(\mathrm{q}(\mathbf{u}_l) \,\|\, \mathrm{p}(\mathbf{u}_l)). \tag{2.89}$$

Evaluating this bound requires $\mathcal{O}(NM^2D)$ computations. Through local reparametrization [70], the gradients of the variational bound can be sampled directly. Evaluations of the DSVI-bound and its gradients have two unbiased sources of stochasticity [94]. First, the expected likelihood is approximated through Monte Carlo samples and second, since the likelihood factorizes along the data, mini-batching is possible. Drawing samples of the likelihood directly elimates the need for Psi-statistics as in (2.84), making DSVI applicable to a broader set of hierarchical Gaussian process models.

Monte Carlo samples of approximate predictions can be drawn using (2.88) as well. The free-form predictive density can be approximated using the mixture

$$\mathrm{q}\big(\mathbf{f}_{L,*} \,\big|\, \mathbf{X}_*\big) \approx \frac{1}{S} \sum_{s=1}^{S} \mathrm{q}\big(\mathbf{q}_{L,*} \,\big|\, \mathbf{u}_L, \mathbf{f}_{L-1,*}\big), \tag{2.90}$$

with S samples propagated through the hierarchical GP starting from the $\mathbf{X}_*$.

Chapter 3

Data Association

The deep Gaussian processes introduced in Section 2.7 are motivated with practical machine learning considerations: It is hard to represent some functions of interest with a standard GP with a smooth prior, most notably functions which have non-smooth components. By repeatedly warping the input space, a deep GP can represent nonstationary structure through the repeated application of smooth functions. In such a deep GP model, only the joint application of all smooth layers is informative as the separation is not driven through domain assumptions or a generative process. In the following, we explore problems for which a hierarchical structure can be formulated and learn explicit and separate posteriors for the different parts of the hierarchy.

Real-world data often include multiple operational regimes of the considered system, for example, a wind-turbine or gas turbine [55]. As an example, consider a model describing the lift resulting from airflow around the wing profile of an airplane as a function of the attack angle. At low values, the lift increases linearly with the attack angle until the wing stalls, and the characteristic of the airflow changes fundamentally. Building a truthful model of such data requires learning two separate models and correctly associating the observed data to each of the dynamical regimes. A similar example arises if our sensors that measure the lift are faulty in a manner such that we either get an accurate reading or a noisy one. Estimating a model in this scenario is often referred to as a data association problem [8, 30]. where we consider the data to have been generated by a mixture of processes. We are interested in factorizing the data into these components.

Figure 3.1 shows an example of faulty sensor data, where sensor readings are disturbed by uncorrelated and asymmetric noise. Applying standard machine learning approaches to such data can lead to model pollution, where the expressive power of the model is used to explain noise instead of the underlying signal. Solving the data association problem by factorizing the data into signal and noise gives rise to a principled approach to avoid this behavior.

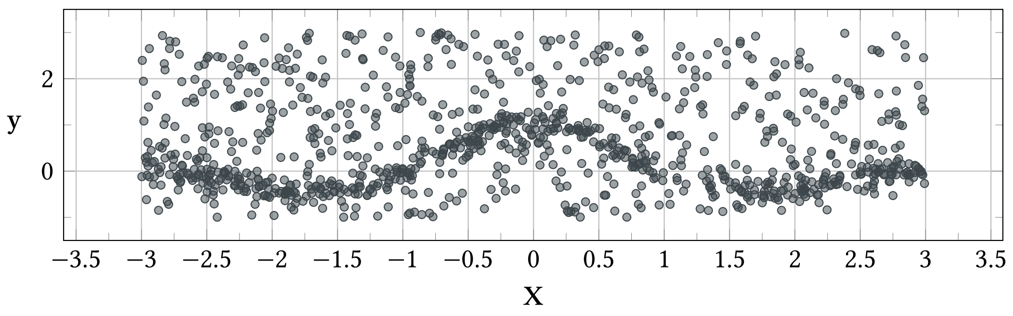


Figure 3.1: A data association problem consisting of two generating processes, one of which is a signal we wish to recover and one is an uncorrelated noise process.

Early approaches to explaining data using multiple generative processes are based on separating the input space and training local expert models explaining easier sub-tasks [61, 86, 115]. The assignment of data points to local experts is handled by a gating network, which learns a function from the inputs to assignment probabilities. However, it is still a central assumption of these models that at every position in the input space, exactly one expert should explain the data. Another approach is presented in [12], where the multimodal regression tasks are interpreted as a density estimation problem. A high number of candidate distributions is reweighed to match the observed data without modeling the underlying generative process.

In contrast, we are interested in a generative process, where data at the same location in the input space could have been generated by a number of global independent processes. Inherently, the data association problem is ill-posed and requires assumptions on both the underlying functions and the association of the observations. In [75] the authors place GP priors on the different generative processes which are assumed to be relevant globally. The associations are modeled via a latent association matrix, and inference is carried out using an expectation-maximization algorithm. This approach takes both the inputs and the outputs of the training data into account to solve the association problem. A drawback is that the model cannot give a posterior estimate about the relevance of the different generating processes at different locations in the input space. This means that the model can be used for data exploration, but additional information is needed to perform predictive tasks. Another approach in [15] expands this model by allowing interdependencies between the different generative processes and formulating the association problem as an inference problem on a latent space and a corresponding covariance function. However, in this approach, the number of components is a free parameter and is prone to overfitting, as the model has no means of turning off components.

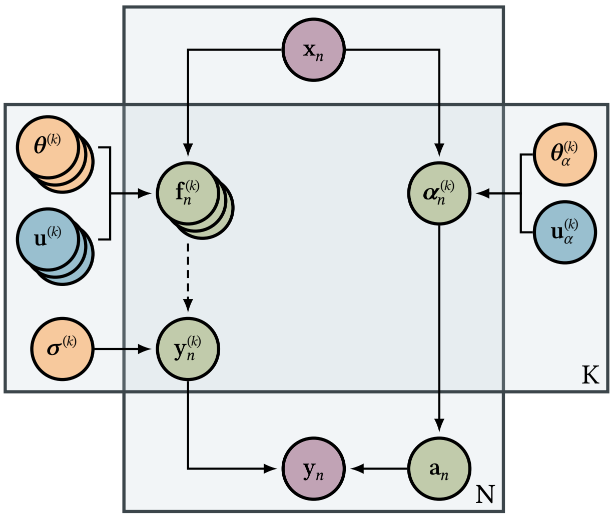

$\mathbf{x}_n$
$\boldsymbol{\theta}^{(k)}$
$\mathbf{u}^{(k)}$
$\mathbf{f}_n^{(k)}$
$\boldsymbol{\alpha}_n^{(k)}$
$\boldsymbol{\theta}_\alpha^{(k)}$
$\mathbf{u}_\alpha^{(k)}$
$\boldsymbol{\sigma}^{(k)}$
$\mathbf{y}_n^{(k)}$
K
$\mathbf{y}_n$
$\mathbf{a}_n$
N

Figure 3.2: The graphical model of DAGP. The violet observations $(\mathbf{x}_n, \mathbf{y}_n)$ are generated by the latent process (green). Exactly one of the K latent functions $f^{(k)}$ and likelihood $\mathbf{y}_n^{(k)}$ are evaluated to generate $\mathbf{y}_n$. We can place shallow or deep GP priors on these latent function values $\mathbf{f}_n^{(k)}$. The assignment $\mathbf{a}_n$ to a latent function is driven by input-dependent weights $\boldsymbol{\alpha}_n^{(k)}$ which encode the relevance of the different functions at $\mathbf{x}_n$. The different parts of the model are determined by the hyperparameters $\boldsymbol{\theta}, \boldsymbol{\sigma}$ (yellow) and variational parameters $\mathbf{u}$ (blue).

In this chapter, we formulate a Bayesian model for the data association problem. Underpinning our approach is the use of GP priors that encode structure both on the functions and the associations themselves, allowing us to incorporate the available prior knowledge about the proper factorization into the learning problem. The use of GP priors allows us to achieve principled regularization without reducing the solution space leading to a well-regularized learning problem. Importantly, we simultaneously solve the association problem for the training data taking both inputs and outputs into account while also obtaining posterior belief about the relevance of the different generating processes in the input space. Our model can describe non-stationary processes in the sense that a different number of processes can be activated in different locations in the input space. We describe this non-stationary structure using additional GP priors, which allows us to make full use of problem-specific knowledge. This leads to a flexible yet interpretable model with a principled treatment of uncertainty.

3.1 Data Association with Gaussian Processes

The data association with Gaussian processes (DAGP) model assumes that there exist K independent functions $\left\{f^{(k)}\right\}_{k=1}^{K}$, which generate pairs of observations $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$. Each data point is generated by evaluating one of the K latent functions and adding Gaussian noise from a corresponding likelihood. The assignment of the n^{th} data point to one of the functions is specified by the indicator vector $\mathbf{a}_n \in \{0, 1\}^K$, which has exactly one non-zero entry. Our goal is to formulate simultaneous Bayesian inference on the functions $f^{(k)}$ and the assignments $\mathbf{a}_n$.

For notational conciseness, we follow the GP related notation in [59] and collect all N inputs as $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ and all outputs as $\mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_N)$. We further denote the k^{th} latent function value associated with the n^{th} data point as $\mathbf{f}_n^{(k)} = f^{(k)}(\mathbf{x}_n)$ and collect them as $\mathbf{F}^{(k)} = \left(\mathbf{f}_1^{(k)}, \dots, \mathbf{f}_N^{(k)}\right)$ and $\mathbf{F} = \left(\mathbf{F}^{(1)}, \dots, \mathbf{F}^{(K)}\right)$. We refer to the k^{th} entry in $\mathbf{a}_n$ as $a_n^{(k)}$ and denote $\mathbf{A} = (\mathbf{a}_1, \dots, \mathbf{a}_N)$.

Given this notation, the marginal likelihood of DAGP can be separated into the likelihood, the latent function processes, and the assignment process and is given by,

$$\begin{aligned} \mathrm{p}(\mathbf{Y} \mid \mathbf{X}) &= \int \mathrm{p}(\mathbf{Y} \mid \mathbf{F}, \mathbf{A})\, \mathrm{p}(\mathbf{F} \mid \mathbf{X})\, \mathrm{p}(\mathbf{A} \mid \mathbf{X}) \, \mathrm{d}\mathbf{A} \, \mathrm{d}\mathbf{F} \\ \mathrm{p}(\mathbf{Y} \mid \mathbf{F}, \mathbf{A}) &= \prod_{n=1}^{N} \prod_{k=1}^{K} \mathcal{N}\left(\mathbf{y}_n \,\middle|\, \mathbf{f}_n^{(k)}, \left(\sigma^{(k)}\right)^2\right)^{\mathbb{I}(a_n^{(k)}=1)}, \end{aligned} \tag{3.1}$$

where $\sigma^{(k)}$ is the noise of the k^{th} Gaussian likelihood and $\mathbb{I}$ is the indicator function.

Since we assume the K processes to be independent given the data and assignments, we place independent GP priors on the latent functions

$$\mathrm{p}(\mathbf{F} \mid \mathbf{X}) = \prod_{k=1}^{K} \mathcal{N}\left(\mathbf{F}^{(k)} \,\middle|\, \mu^{(k)}(\mathbf{X}), \mathrm{k}^{(k)}(\mathbf{X}, \mathbf{X})\right) \tag{3.2}$$

with mean function $\mu^{(k)}$ and kernel $\mathrm{k}^{(k)}$. Our prior on the assignment process is composite. First, we assume that the $\mathbf{a}_n$ are drawn independently from multinomial distributions with logit parameters $\boldsymbol{\alpha}_n = \left(\alpha_n^{(1)}, \dots, \alpha_n^{(K)}\right)$. One approach to specifying $\boldsymbol{\alpha}_n$ is to assume them to be known a priori and to be equal for all data points [75]. Instead, we want to infer them from the data. Specifically, we assume that there is a relationship between the location in the input space $\mathbf{x}$ and the associations. By placing independent GP priors on $\boldsymbol{\alpha}^{(k)}$, we can encode our prior knowledge of the associations via the choice of the covariance function $\mathrm{p}(\boldsymbol{\alpha}|\mathbf{X}) = \prod_{k=1}^{K} \mathcal{N}\left(\boldsymbol{\alpha}^{(k)} \middle| \mathbf{0}, \mathrm{k}_{\alpha}^{(k)}(\mathbf{X}, \mathbf{X})\right)$. The prior on the assignments $\mathbf{A}$ is given by marginalizing the $\boldsymbol{\alpha}^{(k)}$, which, when normalized, parametrize a batch of multinomial distributions,

$$\mathrm{p}(\mathbf{A} \mid \mathbf{X}) = \int \mathcal{M}(\mathbf{A} \mid \operatorname{softmax}(\boldsymbol{\alpha}))\, \mathrm{p}(\boldsymbol{\alpha} \mid \mathbf{X})\, \mathrm{d}\boldsymbol{\alpha}. \tag{3.3}$$

Modelling the relationship between the input and the associations allows us to efficiently model data, which, for example, is unimodal in some parts of the input space and bimodal in others. A simple smoothness prior will encode a belief for how quickly the components switch across the input domain.

Since the GPs of the $\boldsymbol{\alpha}^{(k)}$ use a zero mean function, our prior assumption is a uniform distribution of the different generative processes everywhere in the input space. If inference on the $\mathbf{a}_n$ reveals that, say, all data points at similar positions in the input space can be explained by the same k^{th} process, the belief about $\boldsymbol{\alpha}$ can be adjusted to make a non-uniform distribution favorable at this position, thereby increasing the likelihood via $\mathrm{p}(\mathbf{A} \mid \mathbf{X})$. This mechanism introduces an incentive for the model to use as few functions as possible to explain the data, and, importantly, allows us to predict the relative importance of these functions when calculating the posterior of the new observations $\mathbf{x}_*$.

Figure 3.2 shows the resulting graphical model, which divides the generative process for every data point in the application of the latent functions on the left side and the assignment process on the right side. The interdependencies between the data points are introduced through the GP priors on $\mathbf{f}_n^{(k)}$ and $\boldsymbol{\alpha}_n^{(k)}$ and depend on the hyperparameters $\boldsymbol{\theta} = \left\{\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}_{\alpha}^{(k)}, \sigma^{(k)}\right\}_{k=1}^{K}$. The priors for the $f^{(k)}$ can be chosen independently to encode different prior assumptions about the underlying processes. In Section 3.3, we use different kernels to separate a non-linear signal from a noise process. Going further, we can also use deep GP as priors for the $f^{(k)}$ [32, 94].

3.2 Variational Approximation

Exact inference is intractable in this model. Instead, we formulate a variational approximation based on the doubly stochastic variational inference scheme from Section 2.7. Because of the rich structure in our model, finding a variational lower bound which is both faithful and can be evaluated analytically is hard. To proceed, we formulate an approximation which factorizes along both the K processes and N data points. This bound can be sampled efficiently and allows us to optimize both the models for the different processes $\left\{f^{(k)}\right\}_{k=1}^{K}$ and our belief about the data assignments $\{\mathbf{a}_n\}_{n=1}^{N}$ simultaneously using stochastic optimization.

Variational Lower Bound

As discussed in Section 2.6, we augment all GPs in our model using sets of M inducing points $\mathbf{Z}^{(k)} = \left(\mathbf{z}_1^{(k)}, \dots, \mathbf{z}_M^{(k)}\right)$ and their corresponding function values $\mathbf{u}^{(k)} = f^{(k)}\left(\mathbf{Z}^{(k)}\right)$, the inducing variables. We collect them as $\mathbf{Z} = \left\{\mathbf{Z}^{(k)}, \mathbf{Z}_\alpha^{(k)}\right\}_{k=1}^{K}$ and $\mathbf{U} = \left\{\mathbf{u}^{(k)}, \mathbf{u}_\alpha^{(k)}\right\}_{k=1}^{K}$. Taking the function $f^{(k)}$ and its corresponding GP as an example, the inducing variables $\mathbf{u}^{(k)}$ are jointly Gaussian with the latent function values $\mathbf{F}^{(k)}$ of the observed data by the definition of GPs. We follow (2.58) and choose the variational approximation $\mathrm{q}\left(\mathbf{F}^{(k)}, \mathbf{u}^{(k)}\right) = \mathrm{p}\left(\mathbf{F}^{(k)} \,\middle|\, \mathbf{u}^{(k)}, \mathbf{X}, \mathbf{Z}^{(k)}\right) \mathrm{q}\left(\mathbf{u}^{(k)}\right)$ with $\mathrm{q}\left(\mathbf{u}^{(k)}\right) = \mathcal{N}\left(\mathbf{u}^{(k)} \,\middle|\, \mathbf{m}^{(k)}, \mathbf{S}^{(k)}\right)$. This formulation introduces the set $\left\{\mathbf{Z}^{(k)}, \mathbf{m}^{(k)}, \mathbf{S}^{(k)}\right\}$ of variational parameters indicated in Figure 3.2. To simplify notation, we drop the dependency on $\mathbf{Z}$ in the following.

A central assumption of this approximation is that given enough well-placed inducing variables $\mathbf{u}^{(k)}$, they are a sufficient statistic for the latent function values $\mathbf{F}^{(k)}$. This implies conditional independence of the $\mathbf{f}_n^{(k)}$ given $\mathbf{u}^{(k)}$ and $\mathbf{X}$. The variational posterior of a single GP can then be written as,

$$\begin{aligned}\mathrm{q}\left(\mathbf{F}^{(k)} \,\middle|\, \mathbf{X}\right) &= \int \mathrm{q}\left(\mathbf{u}^{(k)}\right) \mathrm{p}\left(\mathbf{F}^{(k)} \,\middle|\, \mathbf{u}^{(k)}, \mathbf{X}\right) \mathrm{d}\mathbf{u}^{(k)} \\ &= \int \mathrm{q}\left(\mathbf{u}^{(k)}\right) \prod_{n=1}^{N} \mathrm{p}\left(\mathbf{f}_n^{(k)} \,\middle|\, \mathbf{u}^{(k)}, \mathbf{x}_n\right) \mathrm{d}\mathbf{u}^{(k)},\end{aligned} \tag{3.4}$$

which can be evaluated analytically, since it is a convolution of Gaussians. This formulation simplifies inference within single GPs. Next, we discuss how to handle the correlations between the different functions and the assignment processes.

Given a set of assignments $\mathbf{A}$, this factorization along the data points is preserved in our model due to the assumed independence of the different functions in (3.1). This independence is lost if the assignments are unknown. In this case, both the (a priori independent) assignment processes and the functions influence each other through data with unclear assignments. Following the ideas of DSVI in Section 2.7, we maintain these correlations between different parts of the model while assuming factorization of the variational distribution. That is, our variational posterior takes the factorized form,

$$\begin{aligned} q(\mathbf{F}, \boldsymbol{\alpha}, \mathbf{U}) &= q\Big(\boldsymbol{\alpha}, \Big\{\mathbf{F}^{(k)}, \mathbf{u}^{(k)}, \mathbf{u}_\alpha^{(k)}\Big\}_{k=1}^{K}\Big) \\ &= \prod_{k=1}^{K}\prod_{n=1}^{N} p\Big(\boldsymbol{\alpha}_n^{(k)} \,\Big|\, \mathbf{u}_\alpha^{(k)}, \mathbf{x}_n\Big)\, q\Big(\mathbf{u}_\alpha^{(k)}\Big) \prod_{k=1}^{K}\prod_{n=1}^{N} p\Big(\mathbf{f}_n^{(k)} \,\Big|\, \mathbf{u}^{(k)}, \mathbf{x}_n\Big)\, q\Big(\mathbf{u}^{(k)}\Big). \end{aligned} \tag{3.5}$$

Our goal is to recover a posterior for both the generating functions and the assignment of data. To achieve this, instead of marginalizing $\mathbf{A}$, we consider the variational joint of $\mathbf{Y}$ and $\mathbf{A}$,

$$q(\mathbf{Y}, \mathbf{A}) = \int p(\mathbf{Y} \,|\, \mathbf{F}, \mathbf{A})\, p(\mathbf{A} \,|\, \boldsymbol{\alpha})\, q(\mathbf{F}, \boldsymbol{\alpha})\, \mathrm{d}\mathbf{F}\, \mathrm{d}\boldsymbol{\alpha}, \tag{3.6}$$

which retains both the Gaussian likelihood of $\mathbf{Y}$ and the multinomial likelihood of $\mathbf{A}$ in (3.3). A lower bound $\mathscr{L}_{\text{DAGP}}$ for the log-joint $\log p(\mathbf{Y}, \mathbf{A} \,|\, \mathbf{X})$ of DAGP is given by,

$$\begin{aligned} \mathscr{L}_{\text{DAGP}} &= \mathbb{E}_{q(\mathbf{F}, \boldsymbol{\alpha}, \mathbf{U})}\left[\log \frac{p(\mathbf{Y}, \mathbf{A}, \mathbf{F}, \boldsymbol{\alpha}, \mathbf{U} \,|\, \mathbf{X})}{q(\mathbf{F}, \boldsymbol{\alpha}, \mathbf{U})}\right] \\ &= \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{f}_n)}[\log p(\mathbf{y}_n \,|\, \mathbf{f}_n, \mathbf{a}_n)] + \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\alpha}_n)}[\log p(\mathbf{a}_n \,|\, \boldsymbol{\alpha}_n)] \\ &\quad - \sum_{k=1}^{K} \mathrm{KL}(q(\mathbf{u}^{(k)}) \,\|\, p(\mathbf{u}^{(k)} \,|\, \mathbf{Z}^{(k)})) - \sum_{k=1}^{K} \mathrm{KL}(q\Big(\mathbf{u}_\alpha^{(k)}\Big) \,\|\, p\Big(\mathbf{u}_\alpha^{(k)} \,\Big|\, \mathbf{Z}_\alpha^{(k)}\Big)). \end{aligned} \tag{3.7}$$

Due to the structure of (3.5), the bound factorizes along the data enabling stochastic optimization. This bound has complexity $\mathcal{O}\big(NM^2K\big)$ to evaluate.

Optimization of The Lower Bound

An important property of the variational bound for DSVI is that taking samples for single data points is straightforward and can be implemented efficiently. Specifically, for some k and n, samples $\mathbf{\hat{f}_n^{(k)}}$ from $q\Big(\mathbf{f}_n^{(k)}\Big)$ are independent of all other parts of the model and can be drawn using samples from univariate unit Gaussians using reparametrization [70, 88].

Note that it would not be necessary to sample from the different processes, since $q\big(\mathbf{F}^{(k)}\big)$ can be computed analytically [56]. However, we apply the sampling scheme to the optimization of both the assignment processes $\boldsymbol{\alpha}$ and the assignments $\mathbf{A}$. For $\boldsymbol{\alpha}$, the analytic propagation of uncertainties through the softmax renormalization and multinomial likelihoods is intractable, but can be evaluated through sampling.

We optimize $\mathcal{L}_{\text{DAGP}}$ to simultaneously recover maximum likelihood estimates of the hyperparameters $\boldsymbol{\theta}$, the variational parameters $\{\mathbf{Z}, \mathbf{m}, \mathbf{S}\}$, and assignments $\mathbf{A}$. For every n, we represent the belief about $\mathbf{a}_n$ as a K-dimensional discrete distribution $\operatorname{q}(\mathbf{a}_n)$. This distribution models the result of drawing a sample from $\mathcal{M}(\mathbf{a}_n \mid \operatorname{softmax}(\boldsymbol{\alpha}_n))$ during the generation of the data point $(\mathbf{x}_n, \mathbf{y}_n)$.

Since we want to optimize $\mathcal{L}_{\text{DAGP}}$ using (stochastic) gradient descent, we need to employ a continuous relaxation to gain informative gradients of the bound with respect to the discrete vectors $\mathbf{a}_n$. One straightforward way to relax the problem is to use the current belief about $\operatorname{q}(\mathbf{a}_n)$ as parameters for a convex combination of the $\mathbf{f}_n^{(k)}$, that is, to approximate $\mathbf{f}_n \approx \sum_{k=1}^K \operatorname{q}\left(\mathbf{a}_n^{(k)}\right)\hat{\mathbf{f}}_n^{(k)}$. Using this relaxation is problematic in practice. Explaining data points as mixtures of the different generating processes violates the modeling assumption that every data point was generated using exactly one function but can substantially simplify the learning problem. Because of this, special care must be taken during optimization to enforce the sparsity of $\operatorname{q}(\mathbf{a}_n)$.

To avoid this problem, we propose using a different relaxation based on additional stochasticity. Instead of directly using $\operatorname{q}(\mathbf{a}_n)$ to combine the $\mathbf{f}_n^{(k)}$, we first draw a sample $\hat{\mathbf{a}}_\mathbf{n}$ from a concrete random variable as suggested by Maddison et al. [77], parameterized by $\operatorname{q}(\mathbf{a}_n)$. Based on a temperature parameter λ, a concrete random variable enforces sparsity but is also continuous and yields informative gradients using automatic differentiation. Samples from a concrete random variable are unit vectors, and for $\lambda \to 0$, their distribution approaches a discrete distribution.

Our approximate evaluation of the bound in (3.7) during optimization has multiple sources of stochasticity, all of which are unbiased. First, we approximate the expectations using Monte Carlo samples $\hat{\mathbf{f}}_n^{(k)}$, $\hat{\boldsymbol{\alpha}}_n^{(k)}$, and $\hat{\mathbf{a}}_n$. And second, the factorization of the bound along the data allows us to use mini-batches for optimization [56, 94].

Approximate Predictions

Predictions for a test location $\mathbf{x}_*$ are mixtures of K independent Gaussians, given by,

$$\operatorname{q}(\mathbf{f}_* \mid \mathbf{x}_*) = \int \sum_{k=1}^K \operatorname{q}\left(a_*^{(k)} \,\middle|\, \mathbf{x}_*\right) \operatorname{q}\left(\mathbf{f}_*^{(k)} \,\middle|\, \mathbf{x}_*\right) \mathrm{d}\mathbf{a}_*^{(k)} \approx \sum_{k=1}^K \hat{a}_*^{(k)} \hat{\mathbf{f}}_*^{(k)}. \tag{3.8}$$

The predictive posteriors of the K functions $\mathrm{q}\left(\mathbf{f}_*^{(k)} \,\middle|\, \mathbf{x}_*\right)$ are given by K independent shallow GPs and can be calculated analytically using (2.62). Samples from the predictive density over $\mathrm{q}(\mathbf{a}_* \,|\, \mathbf{x}_*)$ can be obtained by sampling from the GP posteriors $\mathrm{q}\left(\boldsymbol{\alpha}_*^{(k)} \,\middle|\, \mathbf{x}_*\right)$ and renormalizing the resulting vector $\boldsymbol{\alpha}_*$ using the softmax-function. The distribution $\mathrm{q}(\mathbf{a}_* \,|\, \mathbf{x}_*)$ reflects the model's belief about how many and which of the K generative processes are relevant at the test location $\mathbf{x}_*$ and their relative probability.

Deep Gaussian Processes

For clarity, we have described the variational bound in terms of a shallow GP. However, as long as their variational bound can be sampled efficiently, any model can be used in place of shallow GPs for the $f^{(k)}$. Since our approximation is based on DSVI, an extension to deep GPs is straightforward. Out new prior assumption about the k^{th} latent function values $\mathrm{p}\left(\mathbf{F}'^{(k)} \,\middle|\, \mathbf{X}\right)$ is given by,

$$\mathrm{p}\left(\mathbf{F}'^{(k)} \,\middle|\, \mathbf{X}\right) = \prod_{l=1}^{L} \mathrm{p}\left(\mathbf{F}_l'^{(k)} \,\middle|\, \mathbf{u}_l'^{(k)} \mathbf{F}_{l-1}'^{(k)}, \mathbf{Z}_l'^{(k)}\right), \tag{3.9}$$

for an L-layer deep GP and with $\mathbf{F}_0'^{(k)} \coloneqq \mathbf{X}$. Similar to the single-layer case, we introduce sets of inducing points $\mathbf{Z}_l'^{(k)}$ and a variational distribution over their corresponding function values $\mathrm{q}\left(\mathbf{u}_l'^{(k)}\right) = \mathcal{N}\left(\mathbf{u}_l'^{(k)} \,\middle|\, \mathbf{m}_l'^{(k)}, \mathbf{S}_l'^{(k)}\right)$. We collect the latent multi-layer function values as $\mathbf{F}' = \{\mathbf{F}_l'^{(k)}\}_{k=1,l=1}^{K,L}$ and corresponding $\mathbf{U}'$ and assume an extended variational distribution,

$$\begin{aligned}
\mathrm{q}(\mathbf{F}', \boldsymbol{\alpha}, \mathbf{U}') &= \mathrm{q}\left(\boldsymbol{\alpha}, \left\{\mathbf{u}_\alpha^{(k)}\right\}_{k=1}^{K}, \left\{\mathbf{F}_l'^{(k)}, \mathbf{u}_l'^{(k)}\right\}_{k=1,l=1}^{K,L}\right) \\
&= \prod_{k=1}^{K}\prod_{n=1}^{N} \mathrm{p}\left(\boldsymbol{\alpha}_n^{(k)} \,\middle|\, \mathbf{u}_\alpha^{(k)}, \mathbf{x}_n\right) \mathrm{q}\left(\mathbf{u}_\alpha^{(k)}\right) \prod_{k=1}^{K}\prod_{l=1}^{L}\prod_{n=1}^{N} \mathrm{p}\left(\mathbf{f}_{n,l}'^{(k)} \,\middle|\, \mathbf{u}_l'^{(k)}, \mathbf{x}_n\right) \mathrm{q}\left(\mathbf{u}_l'^{(k)}\right),
\end{aligned} \tag{3.10}$$

where we identify $\mathbf{f}_n'^{(k)} = \mathbf{f}_{n,L}'^{(k)}$. As the n^{th} marginal of the L^{th} layer depends only on the n^{th} marginal of all layers above sampling from them remains straightforward [94]. The marginal is given by,

$$\mathrm{q}(\mathbf{f}_{n,L}'^{(k)}) = \int \mathrm{q}(\mathbf{f}_{n,L}'^{(k)} \,|\, \mathbf{f}_{n,L-1}'^{(k)}) \prod_{l=1}^{L-1} \mathrm{q}(\mathbf{f}_{n,l}'^{(k)} \,|\, \mathbf{f}_{n,l-1}'^{(k)}) \,\mathrm{d}\mathbf{f}_{n,l}'^{(k)}. \tag{3.11}$$

Table 3.1: Comparison of qualitative model capabilities. A model has a capability if it contains components which could, in principle, solve the respective task.

	Predictive Posterior	Multi-modal Data	Scalable Inference	Inter-pretable Priors	Data Asso-ciation	Predictive Assoc.	Separate Models
Experiment					Table 3.2	Table 3.3	Figure 3.4
DAGP	✓	✓	✓	✓	✓	✓	✓
OMGP [75]	✓	✓	–	✓	✓	–	✓
RGPR [86]	✓	✓	–	✓	–	–	–
MLE [115]	✓	–	–	✓	–	–	–
LatentGP [15]	✓	✓	–	✓	–	–	–
GPR [87]	✓	–	✓	✓	–	–	–
BNN+LV [41]	✓	✓	✓	–	–	–	–
MDN [12]	✓	✓	✓	–	–	–	–
MLP	✓	–	✓	–	–	–	–

The complete bound is structurally similar to (3.7) and given by,

$$\begin{aligned}\mathscr{L}'_{\text{DAGP}} &= \sum_{n=1}^{N} \mathbb{E}_{\mathrm{q}(\mathbf{f}'_{\mathbf{n}})}[\log \mathrm{p}(\mathbf{y}_n \mid \mathbf{f}'_{\mathbf{n}}, \mathbf{a}_n)] + \sum_{n=1}^{N} \mathbb{E}_{\mathrm{q}(\boldsymbol{\alpha}_n)}[\log \mathrm{p}(\mathbf{a}_n \mid \boldsymbol{\alpha}_n)] \\ &\quad - \sum_{k=1}^{K} \sum_{l=1}^{L} \mathrm{KL}(\mathrm{q}(\mathbf{u}_l^{(k)}) \,\|\, \mathrm{p}(\mathbf{u}_l^{(k)} \mid \mathbf{Z}_l^{(k)})) - \sum_{k=1}^{K} \mathrm{KL}(\mathrm{q}\left(\mathbf{u}_\alpha^{(k)}\right) \,\|\, \mathrm{p}\left(\mathbf{u}_\alpha^{(k)} \middle| \mathbf{Z}_\alpha^{(k)}\right)).\end{aligned} \tag{3.12}$$

To calculate the first term, samples have to be propagated through the deep GP structures. This extended bound thus has complexity $\mathcal{O}(NM^2LK)$ to evaluate in the general case and complexity $\mathcal{O}(NM^2 \cdot \max(L, K))$ if the assignments $\mathbf{a}_n$ take binary values.

3.3 Experiments

In this section, we investigate the behavior of the DAGP model. Table 3.1 compares qualitative properties of DAGP and related work. All models can solve standard regression problems and yield unimodal predictive distributions or, in case of multi-layer perceptrons (MLP), a single point estimate. Both standard Gaussian process regression (GPR) and MLP do not impose structure which enables the models to handle multi-modal data. Mixture density networks (MDN) [12] and the infinite mixtures of Gaussian processes (RGPR) [86] model yield multi-modal posteriors through mixtures with many components but do not solve an association problem. Similarly, Bayesian neural networks with added latent variables (BNN+LV) [41] represent such a mixture through a continuous latent variable. Both the overlapping mixtures of Gaussian processes (OMGP) [75] model and DAGP explicitly model the data association problem and yield independent models for the different generating processes. However, OMGP assumes global relevance of the different modes. In contrast, DAGP infers a spacial posterior of this relevance. We evaluate our model on three problems to highlight the following advantages of the explicit structure of DAGP:

Interpretable priors give structure to ill-posed data association problems. First, we consider a noise separation problem, where a signal of interest is disturbed with uniform noise. To solve this problem, assumptions about what constitutes a signal are needed. The hierarchical structure of DAGP allows us to formulate independent and interpretable priors on the noise and signal processes.

Predictive associations represent knowledge about the relevance of generative processes. We then investigate the implicit incentive of DAGP to explain data using as few processes as possible. Additional to a joint posterior explaining the data, DAGP also gives insight into the relative importance of the different processes in different parts of the input space. DAGP can recover the changing number of modes in a data set explicitly.

Separate models for independent generating processes avoid model pollution. Lastly, we simulate a system with multiple operational regimes via mixed observations of two different cart-pole systems. DAGP successfully learns an informative joint posterior by solving the underlying association problem. We show that the DAGP posterior contains two separate models for the two original operational regimes.

Noise Separation

We consider an experiment based on a noise separation problem. We apply DAGP to a one-dimensional regression problem with uniformly distributed asymmetric outliers in the training data. We use a task proposed by Choi et al. [23] where we sample $x \in [-3, 3]$ uniformly and apply the function $f(x) = (1 - \delta)(\cos(\pi/2 \cdot x)\exp(-(x/2)^2) + \gamma) + \delta \cdot \epsilon$, where $\delta \sim \mathcal{B}(\lambda)$, $\epsilon \sim \mathbb{U}(-1, 3)$ and $\gamma \sim \mathcal{N}(0, 0.15^2)$. That is, a fraction λ of the training data, the outliers, are replaced by asymmetric uniform noise. We sample a total of 1000 data points and use 25 inducing points for every GP in our model.

Every generating process in our model can use a different kernel and therefore encode different prior assumptions. For this setting, we use two processes, one with a squared exponential kernel and one with a white noise kernel. This choice encodes the problem statement that every data point is either part of the signal we wish to recover or uncorrelated noise. To avoid pathological solutions for high outlier ratios, we add a

Table 3.2: Results on the ChoiceNet data set. The gray part of the table shows RMSE results for baseline models from [23]. For our experiments using the same setup, we report RMSE comparable to the previous results together with MLL. Both are calculated based on a test set of 1000 equally-spaced samples of the noiseless underlying function.

Outliers	DAGP	OMGP	DAGP	OMGP	CN	MDN	MLP	GPR	RGPR
	MLL	MLL	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE	RMSE
0 %	**2.86**	2.09	0.008	**0.005**	0.034	0.028	0.039	0.008	0.017
20 %	**2.71**	1.83	0.008	**0.005**	0.022	0.087	0.413	0.280	0.013
40 %	**2.12**	1.60	**0.005**	0.007	0.018	0.565	0.452	0.447	1.322
60 %	0.874	**1.23**	0.031	**0.006**	0.023	0.645	0.636	0.602	0.738
80 %	**0.126**	-1.35	0.128	0.896	**0.084**	0.778	0.829	0.779	1.523

prior to the likelihood variance of the first process, which encodes our assumption that there actually is a signal in the training data.

The model proposed in [23], called ChoiceNet (CN), is a specific neural network structure and inference algorithm to deal with corrupted data. In their work, they compare their approach to the MLP, MDN, GPR, and RGPR models. We add experiments for both DAGP and OMGP. Table 3.2 shows results for outlier rates varied from 0 % to 80 %. Besides the root mean squared error (RMSE) reported in [23], we also report the mean test log-likelihood (MLL).

Since we can encode the same prior knowledge about the signal and noise processes in both OMGP and DAGP, the results of the two models are comparable: For low outlier rates, they correctly identify the outliers and ignore them, resulting in a predictive posterior of the signal equivalent to standard GP regression without outliers. In the special case of 0 % outliers, the models correctly identify that the process modeling the noise is not necessary, thereby simplifying to standard GP regression. For high outlier rates, stronger prior knowledge about the signal is required to still identify it perfectly. Figure 3.3 shows the DAGP posterior for an outlier rate of 60 %. While the function has still been identified well, some of the noise is also explained using this process, thereby introducing slight errors in the predictions.

Multimodal Data

Our second experiment applies DAGP to a multimodal data set. The data, together with recovered posterior attributions, can be seen in Figure 3.4. We uniformly sample

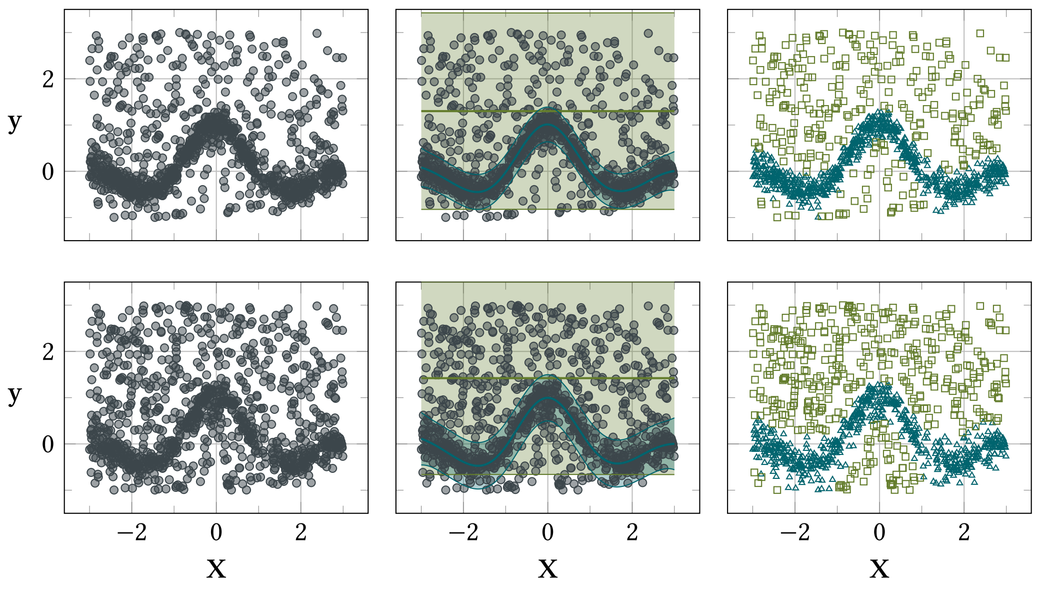


Figure 3.3: DAGP on the ChoiceNet data set with 40 % outliers (upper row) and 60 % outliers (lower row). We show the raw data (left), joint posterior (center) and assignments (right). The bimodal DAGP identifies the signal perfectly up to 40 % outliers. For 60 % outliers, some of the noise is interpreted as signal, but the latent function is still recovered.

350 data points in the interval $x \in [-2\pi, 2\pi]$ and obtain $y_1 = \sin(x) + \epsilon$, $y_2 = \sin(x) - 2\exp(-{}^1\!/_2 \cdot (x-2)^2) + \epsilon$ and $y_3 = -1 - {}^3\!/_{8\pi} \cdot x + {}^3\!/_{10} \cdot \sin(2x) + \epsilon$ with additive independent noise $\epsilon \sim \mathcal{N}(0, 0.005^2)$. The resulting data set $\mathcal{D} = \{(x, y_1), (x, y_2), (x, y_3)\}$ is trimodal in the interval $[0, 5]$ and is otherwise bimodal with one mode containing double the amount of data than the other.

We use squared exponential kernels as priors for both the $f^{(k)}$ and $\alpha^{(k)}$ and 25 inducing points in every GP. Figure 3.4 shows the posterior of a DAGP with $K = 4$ modes applied to the data, which correctly identified the underlying functions. The figure shows the posterior belief about the assignments $\mathbf{A}$ and illustrates that DAGP recovered that it needs only three of the four available modes to explain the data. One of the modes is only assigned points in the interval $[0, 5]$, where the data is trimodal.

This separation is explicitly represented in the model via the assignment processes $\boldsymbol{\alpha}$ (bottom panel in Figure 3.4). Importantly, DAGP does not only cluster the data with respect to the generating processes but also infers a factorization of the input space with respect to the relative importance of the different processes. The model has disabled the mode $k = 2$ in the complete input space and has learned that the mode

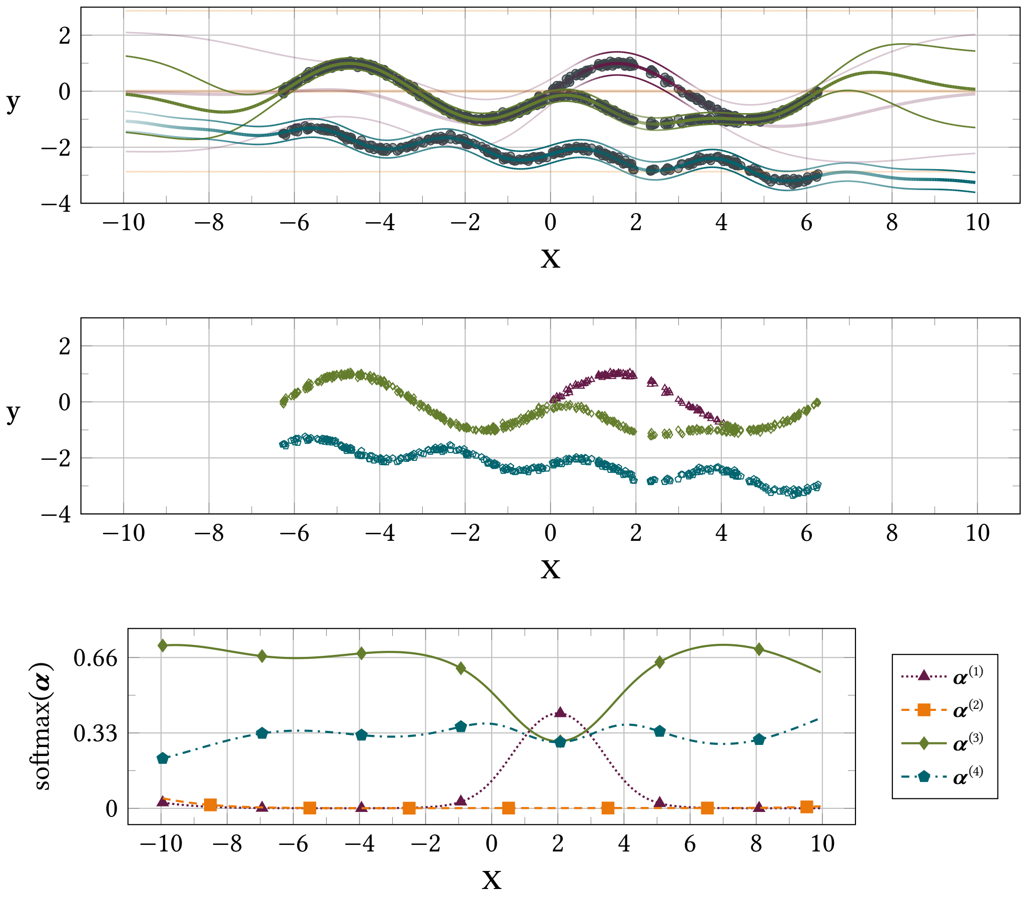


Figure 3.4: The DAGP posterior on an artificial data set with bimodal and trimodal parts. The joint predictions (top) are mixtures of four Gaussians weighed by the assignment probabilities $\boldsymbol{\alpha}$ (bottom). The weights are represented via the opacity of the modes. The model has learned that the mode $k = 2$ is irrelevant, that the mode $k = 1$ is only relevant around the interval $[0, 5]$. Outside this interval, the mode $k = 3$ is twice as likely as the mode $k = 4$. The concrete assignments $\mathbf{a}$ (middle) of the training data show that the mode $k = 1$ is only used to explain observations where the training data is trimodal. The mode $k = 2$ is never used.

Table 3.3: Results on the cart-pole data set. We report mean log likelihoods with their standard error for ten runs. The upper results are obtained by training the model on the mixed data set and evaluating it jointly (left) on multi-modal predictions. We evaluate the two inferred sub-models for the default system (center) and short-pole system (right). We provide gray baseline comparisons with BNN+LV and GPR models which cannot solve the data assignment problem. BNN+LV yields joint predictions which cannot be separated into sub-models. Specialized GPR models trained the individual training sets give a measure of the possible performance if the data assignment problem would be solved perfectly.

	Mixed		Default only	Short-pole only
	Train	Test	Test	Test
DAGP	**0.575 ± 0.013**	**0.521 ± 0.009**	0.844 ± 0.002	**0.602 ± 0.005**
DAGP 2	0.548 ± 0.012	**0.519 ± 0.008**	**0.859 ± 0.001**	0.599 ± 0.011
DAGP 3	0.527 ± 0.004	0.491 ± 0.003	0.852 ± 0.002	0.545 ± 0.012
OMGP	−1.04 ± 0.02	−1.11 ± 0.03	0.66 ± 0.02	−0.81 ± 0.12
BNN+LV	0.519 ± 0.005	0.524 ± 0.005	—	—
GPR Mixed	0.452 ± 0.003	0.421 ± 0.003	—	—
GPR Default	—	—	0.867 ± 0.001	−7.54 ± 0.14
GPR Short	—	—	−5.14 ± 0.04	0.792 ± 0.003

$k = 1$ is only relevant in the interval $[0, 5]$, where the three enabled modes explain about a third of the data each. Outside this interval, the model has learned that one of the modes has about twice the assignment probability than the other one, thus correctly reconstructing the true generative process. The DAGP is implicitly incentivized to explain the data using as few modes as possible through the likelihood term of the inferred $\mathbf{a}_n$ in (3.7). At $x = -10$, the inferred modes and assignment processes start reverting to their respective priors away from the data.

Mixed Cart-Pole Systems

Our third experiment is based on the cart-pole benchmark for reinforcement learning as described by Barto et al. [9] and implemented in OpenAI Gym [20]. In this benchmark, the objective is to apply forces to a cart moving on a frictionless track to keep a pole, which is attached to the cart via a joint, in an upright position. We consider the regression problem of predicting the change of the pole's angle given the current state of the cart and the action applied. The current state of the cart consists of the cart's position and velocity and the pole's angular position and velocity. To simulate a dynamical system with changing system characteristics our experimental setup is to sample trajectories from two different cart-pole systems and merging the resulting data into one training set. The task is not only to learn a model which explains this data well, but to solve the association problem introduced by the different system configurations. This task is important in reinforcement learning settings where we study systems with multiple operational regimes.

We sample trajectories from the system by initializing the pole in an almost upright position and then applying ten uniform random actions. We add Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.01^2)$ to the observed angle changes. To increase the non-linearity of the dynamics, we apply the action for five consecutive time steps and allow the pole to swing freely instead of ending the trajectory after reaching a specific angle. The data set consists of 500 points sampled from the default cart-pole system and another 500 points sampled from a short-pole cart-pole system in which we halve the mass of the pole to 0.05 and shorten the pole to 0.1, a tenth of its default length. This short-pole system is more unstable, and the pole reaches higher speeds. Predictions in this system, therefore, have to take the multimodality into account, as mean predictions between the more stable and the more unstable system can never be observed. We consider three test sets, one sampled from the default system, one sampled from the short-pole system, and a mixture of the two. They are generated by sampling trajectories with an aggregated size of 5000 points from each system for the first two sets and their concatenation for the mixed set.

For this data set, we use squared exponential kernels for both the $f^{(k)}$ and $\alpha^{(k)}$ and 100 inducing points in every GP. We evaluate the performance of deep GPs with up to three layers and squared exponential kernels as models for the different functions. As described in [65, 94], we use identity mean functions for all but the last layers and initialize the variational distributions with low covariances. We compare our models with OMGP and three-layer relu-activated Bayesian neural networks with added latent variables (BNN+LV). The latent variables can be used to effectively model multimodalities and stochasticity in dynamical systems for model-based reinforcement learning [40]. We also compare DAGP to three kinds of sparse GPs (GPR) [59]. They are trained on the mixed data set, the default system, and the short-pole system, and serve as a baseline comparison as these models cannot handle multi-modal data.

Table 3.3 shows results for ten runs of these models. The GPR model predicts a unimodal posterior for the mixed data set that covers both systems. Its mean prediction is approximately the mean of the two regimes and is physically implausible. The DAGP and BNN+LV models yield informative multi-modal predictions with comparable

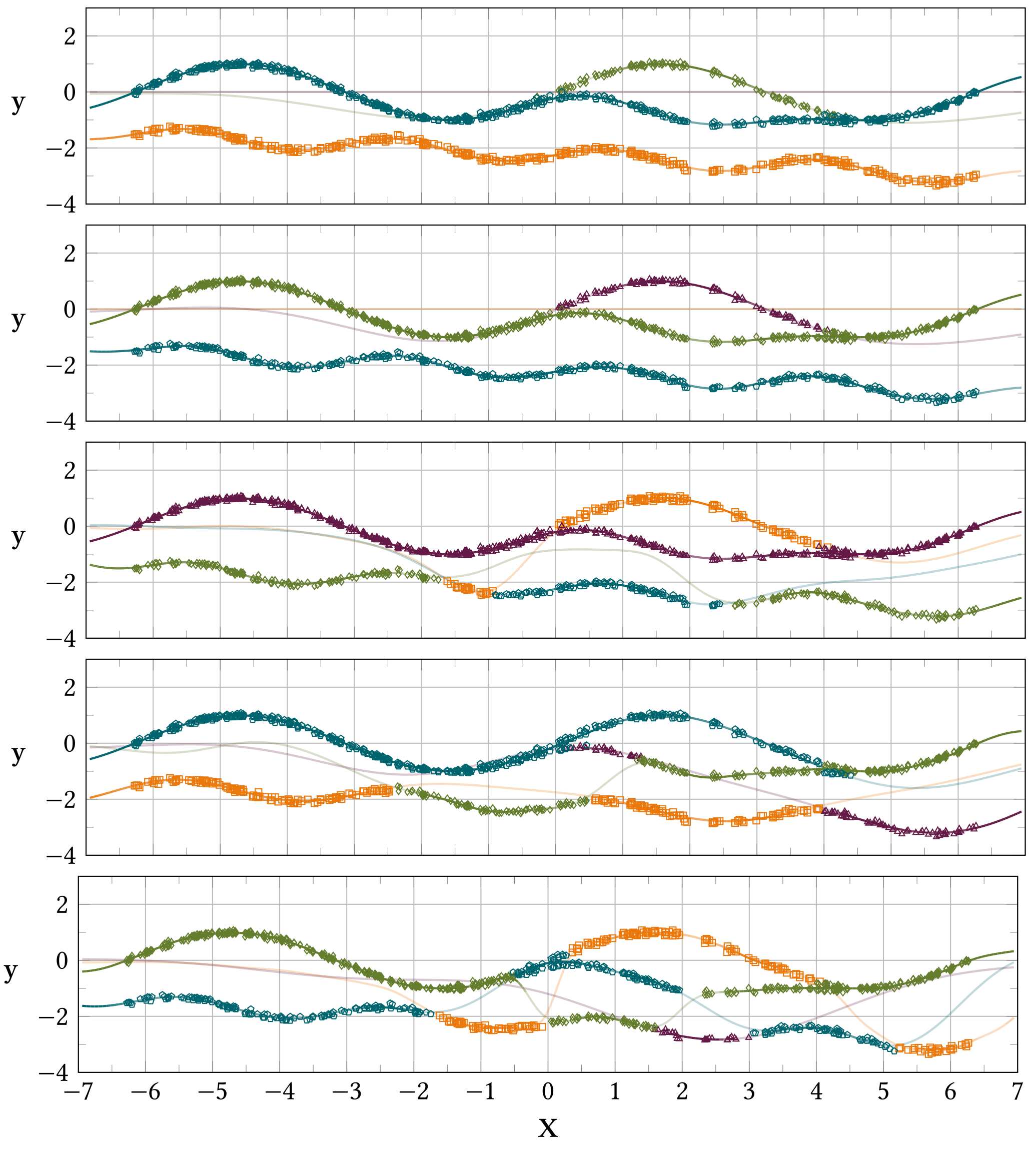


Figure 3.5: Different solutions to the multimodal experiment obtained through retraining. All models explain the data equally well and are plausible under the prior. The upper two models can be considered more interpretable because their solution to the data association problem is more intuitive for experts.

performance. In our setup, OMGP could not successfully solve the data association problem and thus does not produce a useful joint posterior. The OMGP's inference scheme is tailored to ordered one-dimensional problems. It does not trivially translate to the 4D cart-pole problem.

As BNN+LV does not explicitly solve the data association problem, the model does not yield sub-models for the two different systems. Similar results would be obtained with the MDN and RGPR models, which also cannot be separated into sub-models. OMGP and DAGP yield such sub-models which can independently be used for predictions in the default or short-pole systems. Samples drawn from these models can be used to generate physically plausible trajectories in the respective system. OMGP fails to model the short-pole system but does yield a viable model for the default system that evolves more slowly due to higher torque and is therefore easier to learn. In contrast, the two sub-models inferred by DAGP perform well on their respective systems, showing that DAGP reliably solves the data association problem and successfully avoids model pollution by separating the two systems well. Given this separation, shallow and deep models for the two modes show comparable performance. The more expressive deep GPs model the default system slightly better while sacrificing performance on the more difficult short-pole system.

3.4 Discussion

We have presented a fully Bayesian model for the data association problem. Our model factorizes the observed data into a set of independent processes and provides a model for both the processes and data assignments. The data association problem is inherently ill-constrained and requires significant assumptions to recover a solution. We make use of interpretable GP priors allowing global a priori information to be included in the model. Importantly, our model can exploit information both about the underlying functions and the association structure. We have derived a principled approximation to the marginal likelihood, which allows us to perform inference for flexible hierarchical processes.

We have shown how explicitly formulating the hierarchical generative process of the data association problem and deriving explicit posteriors for the different components leads to an expressive model. This model allows us to formulate expert knowledge about different modes or mode associations to facilitate training. At the same time, predictions of model components can be used to solve a multitude of tasks in Table 3.1, which cannot be solved by models that collapse parts of the generative process.

In the noise separation task, data is assigned to two modes which differ on a qualitative level: The noise mode with high uncertainty and a constant prediction can explain any data point, but at the same time, data points have a low likelihood. The mode representing the function of interest can be placed where data is densest and explains the data with high confidence and thus high likelihood. The qualitative difference also allows to model to recognize if there is no noise at all, as explaining all data using the more informative mode is always beneficial.

The multimodal experiments show that the model still performs well when more ambiguity is introduced. In this case, all modes share the same prior and are interchangeable. Due to the incentive to use as few modes as possible, DAGP still correctly identifies that three modes are enough to explain the data in Figure 3.4. More specifically, at any point $\mathbf{X}$, at most three modes are active. Due to the RBF GP prior, the functions identified by the three modes vary slowly and thus, data close together tends to be explained by the same mode.

The deep GP variational approximation and thus the derived approximation for DAGP assumes independence between the layers. An important effect of this assumption is that a DAGP posterior cannot represent interchangeable modes since this would require dependence between $\mathbf{F}$ and $\boldsymbol{\alpha}$. Instead, early on during training of a DAGP instance, the model (randomly) assigns modes of the model $\mathbf{f}^{(k)}$ to modes in the underlying data. Figure 3.4 shows one such association, Figure 3.5 shows other results obtained through retraining. All models explain the data equally well and are plausible under the prior that prefers no more than 3 smooth modes to be active for any $\mathbf{X}$, which is the case for all models. The predictive posterior $\mathrm{p}(\mathbf{f}_* \mid \mathbf{x}_*)$ is comparable for all models which perform equally well with respect to test metrics. However, the lower three models in Figure 3.5 could be considered less interpretable or less desirable because data which intuitively should belong to the same mode does not.

In the DAGP model, these qualitatively different solutions cannot be distinguished automatically but can only be identified through inspection by experts. While the predictive posterior close to the data is similar, the different solutions could generalize differently, introducing a subjective choice of what constitutes the *correct* model. In the next chapter, we explore how this ambiguity can be removed by introducing a prior on the dependence between modes through structured informed by expert knowledge.

Chapter 4

Non-Linear Time-Series Alignment

In Chapter 3, we used hierarchically structured Gaussian processes to solve a data association problem. We exploited the conditional independence between modes given associations to formulate an efficient variational approximation. The dependence between modes introduced through uncertain associations was handled through components that learn expected associations and an explicit posterior for the training data. However, while the different modes in the DAGP model are independent in the true generative process, they might not be independent in a Bayesian posterior due to insufficient information. Assuming independence anyway is the core simplifying assumption in the variational approximation. In this chapter, we explore how dependencies in multimodal models can be represented explicitly in a hierarchical structure.

Many real-world systems are inherently hierarchical and connected. Ideally, a machine learning method should model and recognize such dependencies. Take wind power production, which is one of the major providers for renewable energy today, as an example: To optimize the efficiency of a wind-turbine, the speed and pitch have to be controlled according to the local wind conditions (speed and direction). In a wind-farm, turbines are typically equipped with sensors for wind speed and direction. The goal is to use these sensor data to produce accurate estimates and forecasts of the wind conditions at every turbine in the farm. For the ideal case of a homogeneous and very slowly changing wind field, the wind conditions at each geometrical position in a wind-farm can be estimated using the propagation times (time warps) computed from geometry, wind speed, and direction [14, 95, 106]. In the real world, however, wind fields are not homogeneous, exhibit global and local turbulence, and interfere with the turbines and the terrain inside and outside the farm and further, sensor faults may lead to data loss. This makes it extremely difficult to construct accurate analytical models of wind propagation in a farm. Also, standard approaches for extracting such information from data, for example generalized time warping [125], fail at this task because they rely on a high signal to noise ratio. Instead, we want to construct Bayesian nonlinear dynamic data-based models for wind conditions and warpings which handle the stochastic nature of the system in a principled manner.

In this chapter, we look at a generalization of this type of problem and propose a novel Bayesian approach to finding nonlinear alignments of time-series based on latent shared information. We view the power production of different wind-turbines as the outputs of a multi-output Gaussian process (MO-GP) [4, 17] that models the latent wind fronts. We embed this model in a hierarchy, adding a layer of non-linear alignments on top and a layer of non-linear warpings [74, 102] below which increases flexibility and encodes the original generative process. We show how the resulting model can be interpreted as a group of deep Gaussian processes with the added benefit of covariances between different outputs. The imposed structure is used to formulate prior knowledge in a principled manner, restrict the representational power to physically plausible models, and recover the desired latent wind fronts and relative alignments. The presented model can be interpreted as a group of D deep GPs, all of which share one layer that is a MO-GP. This MO-GP acts as an interface to share information between the different GPs that are otherwise conditionally independent.

4.1 Aligned Multi-Output Gaussian Processes

We are interested in formulating shared priors over a set of functions $\{f_d\}_{d=1}^D$ using GPs, thereby directly parameterizing their interdependencies. In a traditional GP setting, multiple outputs are considered conditionally independent given the inputs, which significantly reduces the computational cost but also prevents the utilization of shared information. Such interdependencies can be formulated via convolution processes (CPs) as proposed by Alvarez et al. [2], Alvarez et al. [3], and Boyle et al. [18], a generalization of the linear model of coregionalization (LMC) [25, 64]. In the CP framework, the output functions are the result of a convolution of the latent processes w_r with smoothing kernel functions $T_{d,r}$ for each output f_d, defined as

$$f_d(\mathbf{x}) = \sum_{r=1}^{R} \int T_{d,r}(\mathbf{x} - \mathbf{z}) \cdot w_r(\mathbf{z}) \,\mathrm{d}\mathbf{z}. \tag{4.1}$$

In this model, the convolutions of the latent processes generating the different outputs are all performed around the same point $\mathbf{x}$. We generalize this by allowing different alignments of the observations that depend on the position in the input space. This allows us to model the changing relative interaction times for the different latent wind

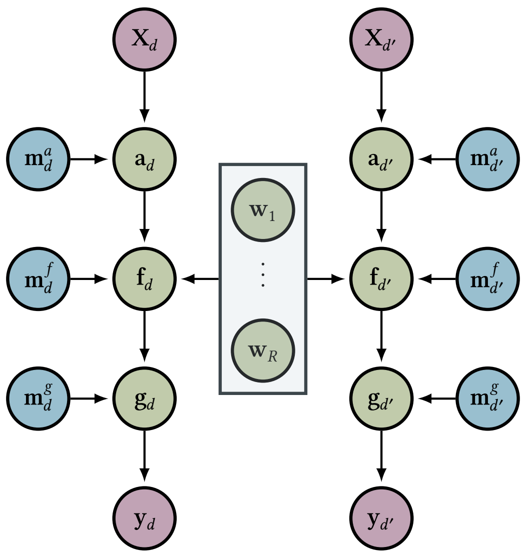


Figure 4.1: The graphical model of AMO-GP. A convolution process, informed by R latent processes, models shared information between multiple data sets with nonlinear alignments and warpings. The shared information connects D deep GPs through a shared layer.

fronts, as described in the introduction. We also assume that the dependent functions f_d are latent themselves, and the data we observe is generated via independent noisy nonlinear transformations of their values. Every function f_d is augmented with an alignment function a_d and a warping g_d on which we place independent GP priors.

For simplicity, we assume that the outputs are evaluated all at the same positions $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^N$. This can easily be generalized to different input sets for every output. In our application, the $\mathbf{x}_n$ are one-dimensional time indices. However, since the model can be generalized to multi-dimensional inputs, we do not restrict ourselves to the one-dimensional case. We note that in the multi-dimensional case, reasoning about priors on alignments can be challenging. We call the observations associated with the d-th function $\mathbf{y}_d$ and use the stacked vector $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_D)$ to collect the data of all outputs. The final model is then given by

$$\mathbf{y}_d = g_d(f_d(a_d(\mathbf{X}))) + \boldsymbol{\epsilon}_d, \tag{4.2}$$

where $\boldsymbol{\epsilon}_d \sim \mathcal{N}(0, \sigma_{y,d}^2 \mathbf{I})$ is a noise term. The functions are applied element-wise. This encodes the generative process described above: For every turbine $\mathbf{y}_d$, observations at positions $\mathbf{X}$ are generated by first aligning to the latent wind fronts using a_d, applying the front in f_d, imposing turbine-specific components g_d and adding noise in $\boldsymbol{\epsilon}_d$.

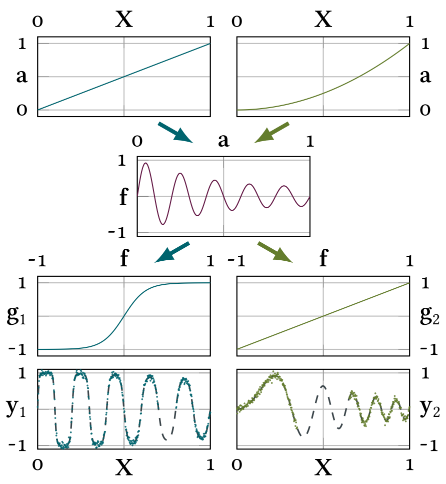


Figure 4.2: An artificial example of hierarchical composite data with multiple observations of shared latent information. This hierarchy generates two data sets using a dampened sine function which is never observed directly.

We assume independence between a_d and g_d across outputs and apply GP priors of the form $a_d \sim \mathcal{GP}(\mathrm{id}, k_{a,d})$ and $g_d \sim \mathcal{GP}(\mathrm{id}, k_{g,d})$. By setting the prior mean to the identity function $\mathrm{id}(x) = x$, the standard CP model is our default assumption. During learning, the model can choose the different a_d and g_d in a way to reveal the independent shared latent processes $\{w_r\}_{r=1}^R$ on which we also place GP priors $w_r \sim \mathcal{GP}(0, k_{u,r})$. Similar to Boyle et al. [18], we assume the latent processes to be independent white noise processes by setting $\mathrm{cov}[w_r(\mathbf{z}), w_{r'}(\mathbf{z}')] = \delta_{rr'}\delta_{\mathbf{z}\mathbf{z}'}$. Under this prior, the f_d are also GPs with zero mean and

$$\mathrm{cov}[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] = \sum_{r=1}^{R} \int T_{d,r}(\mathbf{x} - \mathbf{z}) T_{d',r}(\mathbf{x}' - \mathbf{z}) \, \mathrm{d}\mathbf{z}. \tag{4.3}$$

Using the squared exponential kernel for all $T_{d,r}$, the integral has a closed-form solution. With $\{\sigma_{d,r}, \ell_{d,r}\}$ denoting the kernel hyper parameters associated with $T_{d,r}$, it is given by

$$\mathrm{cov}[f_d(\mathbf{x}), f_{d'}(\mathbf{x}')] = \sum_{r=1}^{R} \frac{(2\pi)^{\frac{K}{2}} \sigma_{d,r} \sigma_{d',r}}{\prod_{k=1}^{K} \hat{\ell}_{d,d',r,k}^{-1}} \exp\left(-\frac{1}{2} \sum_{k=1}^{K} \frac{(x_k - x_k')^2}{\hat{\ell}_{d,d',r,k}^2} \right), \tag{4.4}$$

where $\mathbf{x}$ is K-dimensional and $\hat{\ell}_{d,d',r,k} = \sqrt{\ell_{d,r,k}^2 + \ell_{d',r,k}^2}$.

4.2 Variational Approximation

Since exact inference in this model is intractable, we present a variational approximation to the model's marginal likelihood in this section. Analogously to $\mathbf{y}$, we denote the random vectors which contain the function values of the respective functions and outputs as $\mathbf{a}$ and $\mathbf{f}$. The joint probability distribution of the data can then be written as

$$\begin{aligned}
\mathrm{p}(\mathbf{y}, \mathbf{f}, \mathbf{a} \mid \mathbf{X}) &= \mathrm{p}(\mathbf{f} \mid \mathbf{a}) \prod_{d=1}^{D} \mathrm{p}(\mathbf{y_d} \mid \mathbf{f_d})\, \mathrm{p}(\mathbf{a_d} \mid \mathbf{X}), \text{ with} \\
\mathbf{a_d} \mid \mathbf{X} &\sim \mathcal{N}(\mathbf{X}, \mathbf{K_{a,d}} + \sigma_{a,d}^2 \mathrm{I}), \\
\mathbf{f} \mid \mathbf{a} &\sim \mathcal{N}(\mathbf{0}, \mathbf{K_f} + \sigma_f^2 \mathrm{I}), \\
\mathbf{y_d} \mid \mathbf{f_d} &\sim \mathcal{N}(\mathbf{f_d}, \mathbf{K_{g,d}} + \sigma_{y,d}^2 \mathrm{I}).
\end{aligned} \tag{4.5}$$

Here, we use $\mathbf{K}$ to refer to the Gram matrices corresponding to the respective GPs. All but the convolution processes factorize over the different levels of the model as well as the different outputs.

To approximate a single deep GP, we can use the approximations discussed in Section 2.7. The structure of the AMO-GP model is closely related to a group of stacked GPs with additional dependencies in one of the layers. Since these dependencies are represented with a multi-output GP that, given the latent functions, yields conditionally independent outputs, nested variation compression generalizes to the AMO-GP. The variational lower bound for the AMO-GP is given by

$$\begin{aligned}
\log \mathrm{p}(\mathbf{y} \mid \mathbf{X}, \mathbf{Z}, \mathbf{u}) \geq{} & \sum_{d=1}^{D} \log \mathcal{N}\left(\mathbf{y_d} \,\middle|\, \mathbf{\Psi_{g,d}} \mathbf{K}_{\mathbf{u_{g,d}u_{g,d}}}^{-1} \mathbf{m_{g,d}}, \sigma_{y,d}^2 \mathrm{I}\right) - \sum_{d=1}^{D} \frac{1}{2\sigma_{a,d}^2} \operatorname{tr}(\Sigma_{\mathbf{a,d}}) \\
& - \frac{1}{2\sigma_f^2} \left(\psi_f - \operatorname{tr}\left(\mathbf{\Phi_f} \mathbf{K}_{\mathbf{u_f u_f}}^{-1}\right)\right) - \sum_{d=1}^{D} \frac{1}{2\sigma_{y,d}^2} \left(\psi_{g,d} - \operatorname{tr}\left(\mathbf{\Phi_{g,d}} \mathbf{K}_{\mathbf{u_{g,d}u_{g,d}}}^{-1}\right)\right) \\
& - \sum_{d=1}^{D} \mathrm{KL}(\mathrm{q}(\mathbf{u_{a,d}}) \,\|\, \mathrm{p}(\mathbf{u_{a,d}})) - \mathrm{KL}(\mathrm{q}(\mathbf{u_f}) \,\|\, \mathrm{p}(\mathbf{u_f})) - \sum_{d=1}^{D} \mathrm{KL}(\mathrm{q}(\mathbf{u_{y,d}}) \,\|\, \mathrm{p}(\mathbf{u_{y,d}})) \\
& - \frac{1}{2\sigma_f^2} \operatorname{tr}\left(\left(\mathbf{\Phi_f} - \mathbf{\Psi_f}^{\top} \mathbf{\Psi_f}\right) \mathbf{K}_{\mathbf{u_f u_f}}^{-1} \left(\mathbf{m_f} \mathbf{m_f}^{\top} + \mathbf{S_f}\right) \mathbf{K}_{\mathbf{u_f u_f}}^{-1}\right) \\
& - \sum_{d=1}^{D} \frac{1}{2\sigma_{y,d}^2} \operatorname{tr}\left(\left(\mathbf{\Phi_{g,d}} - \mathbf{\Psi_{g,d}}^{\top} \mathbf{\Psi_{g,d}}\right) \mathbf{K}_{\mathbf{u_{g,d}u_{g,d}}}^{-1} \left(\mathbf{m_{g,d}} \mathbf{m_{g,d}}^{\top} + \mathbf{S_{g,d}}\right) \mathbf{K}_{\mathbf{u_{g,d}u_{g,d}}}^{-1}\right).
\end{aligned} \tag{4.6}$$

The bound contains one Gaussian fit term per output dimension and a series of regularization terms for every GP in the hierarchy. The KL-divergences connect the variational approximations to the prior, and the different trace terms regularize the variances of the different GPs. For a detailed discussion see [57]. This bound depends on the hyper parameters of the kernel and likelihood $\{\ell, \sigma\}$ and the variational parameters $\left\{\mathbf{Z}_{l,d}, \mathbf{m}_{l,d}, \mathbf{S}_{l,d} \middle| l \in \{\mathbf{a}, \mathbf{f}, \mathbf{d}\}, d \in [D]\right\}$.

The bound can be calculated in $\mathcal{O}(NM^2)$ time and factorizes along the data points, enabling stochastic optimization. Since every of the N data points is associated with one of the D outputs, the computational cost of the model is independent of D. Information is only shared between the different outputs using the inducing points in $\mathbf{f}$. As the different outputs share a common function, increasing D allows us to reduce the number of variational parameters per output and still represent the shared information.

A central component of this bound are expectations over kernel matrices, the three Ψ-statistics $\psi_f = \mathbb{E}_{\mathrm{q}(\mathbf{a})}[\mathrm{tr}(\mathbf{K_{ff}})]$, $\mathbf{\Psi_f} = \mathbb{E}_{\mathrm{q}(\mathbf{a})}[\mathbf{K_{fu}}]$ and $\mathbf{\Phi_f} = \mathbb{E}_{\mathrm{q}(\mathbf{a})}[\mathbf{K_{uf}}\mathbf{K_{fu}}]$. Closed-form solutions for these statistics depend on the choice of kernel and are known for specific kernels, such as linear or RBF kernels, for example shown in [32]. In the following subsection, we will give closed-form solutions for these statistics required in the shared CP-layer of our model.

Convolution Kernel Expectations

The uncertainty about the first layer is captured by the variational distribution of the latent alignments $\mathbf{a}$ given by $\mathrm{q}(\mathbf{a}) \sim \mathcal{N}(\boldsymbol{\mu}_a, \Sigma_a)$, with $\mathbf{a} = (\mathbf{a}_1, \dots, \mathbf{a}_d)$. Every aligned point in $\mathbf{a}$ corresponds to one output of $\mathbf{f}$ and ultimately to one of the $\mathbf{y}_d$. Since the closed form of the multi output kernel depends on the choice of outputs, we will use the notation $\hat{f}(\mathbf{a}_n)$ to denote $f_d(\mathbf{a}_n)$ such that $\mathbf{a}_n$ is associated with output d.

For notational simplicity, we only consider the case of one single latent process w_r. Since the latent processes are independent, the results can easily be generalized to multiple processes. Then, ψ_f is given by

$$\begin{aligned}\psi_f &= \mathbb{E}_{\mathrm{q}(\mathbf{a})}[\mathrm{tr}(\mathbf{K_{ff}})] \\ &= \sum_{n=1}^{N} \mathbb{E}_{\mathrm{q}(\mathbf{a}_n)}\left[\mathrm{cov}\left[\hat{f}(\mathbf{a}_n), \hat{f}(\mathbf{a}_n)\right]\right] \\ &= \sum_{n=1}^{N} \hat{\sigma}_{nn}^2.\end{aligned} \tag{4.7}$$

Similar to the notation $\hat{f}(\cdot)$, we use $\hat{\sigma}_{nn'}$ to denote the variance term associated with the covariance function $\operatorname{cov}[\hat{f}(\mathbf{a}_n), \hat{f}(\mathbf{a}_{n'})]$. The expectation $\mathbf{\Psi_f} = \mathbb{E}_{\mathrm{q}(\mathbf{a})}[\mathbf{K_{fu}}]$ connecting the alignments and the pseudo inputs is given by

$$\begin{aligned}
\mathbf{\Psi_f} &= \mathbb{E}_{\mathrm{q}(\mathbf{a})}[\mathbf{K_{fu}}], \text{ with} \\
(\mathbf{\Psi_f})_{ni} &= \int \operatorname{cov}\Big[\hat{f}(\mathbf{a}_n), \hat{f}(\mathbf{Z}_i)\Big]\, \mathrm{q}(\mathbf{a}_n)\, \mathrm{d}\mathbf{a}_n \\
&= \hat{\sigma}_{ni}^2 \sqrt{\frac{(\Sigma_a)_{nn}^{-1}}{\hat{\ell}_{ni} + (\Sigma_a)_{nn}^{-1}}} \cdot \exp\left(-\frac{1}{2}\frac{(\Sigma_a)_{nn}^{-1}\hat{\ell}_{ni}}{(\Sigma_a)_{nn}^{-1} + \hat{\ell}_{ni}}\left((\boldsymbol{\mu}_a)_n - \mathbf{Z}_i\right)^2\right)
\end{aligned} \tag{4.8}$$

where $\hat{\ell}_{ni}$ is the combined length scale corresponding to the same kernel as $\hat{\sigma}_{ni}$. Lastly, $\mathbf{\Phi_f} = \mathbb{E}_{\mathrm{q}(\mathbf{a})}[\mathbf{K_{uf}K_{fu}}]$ connects alignments and pairs of pseudo inputs with the closed form

$$\begin{aligned}
\mathbf{\Phi_f} &= \mathbb{E}_{\mathrm{q}(\mathbf{a})}[\mathbf{K_{uf}K_{fu}}], \text{ with} \\
(\mathbf{\Phi_f})_{ij} &= \sum_{n=1}^{N} \int \operatorname{cov}\Big[\hat{f}(\mathbf{a}_n), \hat{f}(\mathbf{Z}_i)\Big] \cdot \operatorname{cov}\Big[\hat{f}(\mathbf{a}_n), \hat{f}(\mathbf{Z}_j)\Big]\, \mathrm{q}(\mathbf{a}_n)\, \mathrm{d}\mathbf{a}_n \\
&= \sum_{n=1}^{N} \hat{\sigma}_{ni}^2 \hat{\sigma}_{nj}^2 \sqrt{\frac{(\Sigma_a)_{nn}^{-1}}{\hat{\ell}_{ni} + \hat{\ell}_{nj} + (\Sigma_a)_{nn}^{-1}}} \cdot \exp\Bigg(-\frac{1}{2}\frac{\hat{\ell}_{ni}\hat{\ell}_{nj}}{\hat{\ell}_{ni} + \hat{\ell}_{nj}}(\mathbf{Z}_i - \mathbf{Z}_j)^2 \\
&\qquad - \frac{1}{2}\frac{(\Sigma_a)_{nn}^{-1}(\hat{\ell}_{ni} + \hat{\ell}_{nj})}{(\Sigma_a)_{nn}^{-1} + \hat{\ell}_{ni} + \hat{\ell}_{nj}} \cdot \left((\boldsymbol{\mu}_a)_n - \frac{\hat{\ell}_{ni}\mathbf{Z}_i + \hat{\ell}_{nj}\mathbf{Z}_j}{\hat{\ell}_{ni} + \hat{\ell}_{nj}}\right)^2\Bigg).
\end{aligned} \tag{4.9}$$

Note that the Ψ-statistics factorize along the data, and we only need to consider the diagonal entries of Σ_a. If all the data belong to the same output, the Ψ-statistics of the standard squared exponential kernel can be recovered as a special case. It is used to propagate the uncertainties through the output-specific warpings $\mathbf{g}$.

Approximate Predictions

Using the variational lower bound in (4.6), our model can be fitted to data, resulting in appropriate choices of the kernel hyper-parameters and variational parameters. Now assume we want to predict approximate function values $\mathbf{g}_{d,*}$ for previously unseen points $\mathbf{X}_{d,*}$ associated with output d, which are given by $\mathbf{g}_{d,*} = g_d(f_d(a_d(\mathbf{X}_{d,*})))$.

Because of the conditional independence assumptions in the model, other outputs $d' \neq d$ only have to be considered in the shared layer $\mathbf{f}$. In this shared layer, the belief about the different outputs and the shared information and is captured by the variational distribution $q(\mathbf{u}_f)$. Given $q(\mathbf{u}_f)$, the different outputs are conditionally independent of one another, and thus, predictions for a single dimension in our model are equivalent to predictions in a single deep GP with nested variational compression.

4.3 Model Interpretation

The graphical model of the aligned multi-output Gaussian process model (AMO-GP) in Figure 4.1 illustrates that the presented model can be interpreted as a group of D deep GPs all of which share one layer which is a CP. This CP acts as an interface to share information between the different GPs that are otherwise conditionally independent. This modeling-choice introduces a new quality to the model when compared to standard deep GPs with multiple output dimensions since the latter cannot learn dependencies between the different outputs. Compared to standard multi-output GPs, the AMO-GP introduces more flexibility with respect to the shared information. CPs make strong assumptions about the relative alignments of the different outputs, that is, they assume constant time-offsets. The AMO-GP extends this by introducing a principled Bayesian treatment of general nonlinear alignments a_d on which we can place informative priors derived from the problem at hand. Together with the warping layers g_d, our model can learn to share knowledge in an informative latent space learned from the data.

Alternatively, this model can be interpreted as a shared and warped latent variable model with a very specific prior: The indices $\mathbf{X}$ are part of the prior for the latent space $a_d(\mathbf{X})$ and specify a sense of order for the different data points $\mathbf{y}$ which is augmented with uncertainty by the alignment functions. Using this order, the convolution processes enforce the covariance structure for the different datapoints specified by the smoothing kernels.

In order to derive an inference scheme, we need the ability to propagate uncertainties about the correct alignments and latent shared information through subsequent layers. We adapted the approach of nested variational compression by Hensman et al. [57], which is originally concerned with a single deep GP. The approximation is expanded to handle multiple GPs at once, yielding the bound in (4.6). The bound reflects the dependencies of the different outputs as the sharing of information between the different deep GPs is approximated through the shared inducing variables $\mathbf{u}_{f,d}$.

4.4 Experiments

In this section we show how to apply the AMO-GP to the task of finding the common structure in time-series observations. In this setting, we observe multiple time-series $\mathcal{T}_d = (\mathbf{X}_d, \mathbf{y}_d)$ and assume that there exist latent time-series which determine the observations.

We will first apply the AMO-GP to an artificial data set in which we define a decomposed system of dependent time-series by specifying a shared latent function generating the observations together with relative alignments and warpings for the different time-series. We will show that our model can recover this decomposition from the training data and compare the results to other approaches of modeling the data. Then we focus on a real-world data set of a neighboring pair of wind-turbines in a wind-farm, where the model can recover a representation of the latent prevailing wind condition and the relative timings of wind fronts at the two turbines.

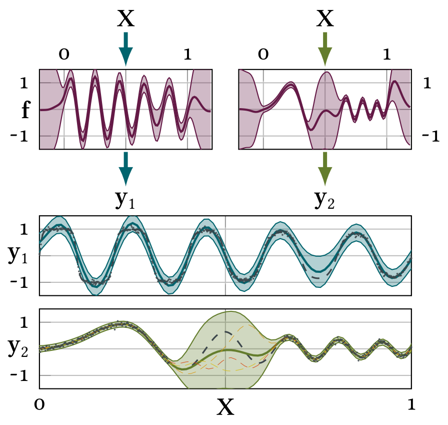
Artificial Data Set

Our data set consists of two time-series $\mathcal{T}_1$ and $\mathcal{T}_2$ generated by a dampened sine function.

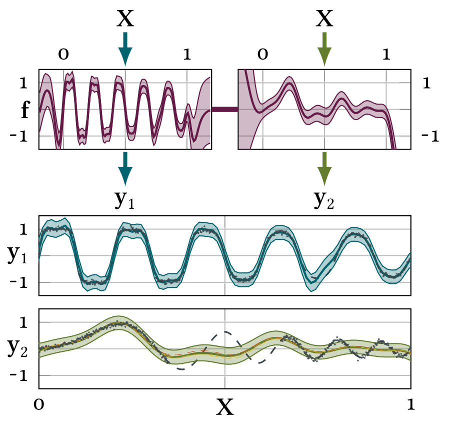
$$f : \begin{cases} [0,1] \to \mathbb{R} \\ \quad x \mapsto \left(1 - \frac{3}{4} \tanh\left(\frac{10\pi}{15} \cdot x\right)\right) \cdot \sin(10\pi \cdot x). \end{cases} \tag{4.10}$$

We choose the alignment of $\mathcal{T}_1$ and the warping of $\mathcal{T}_2$ to be the identity in order to prevent us from directly observing the latent function and apply a sigmoid warping to $\mathcal{T}_1$. The alignment of $\mathcal{T}_2$ is selected to be a quadratic function. Figure 4.2 shows a visualization of this decomposed system of dependent time-series. To obtain training data, we uniformly sampled 500 points from the two time-series and added Gaussian noise. We subsequently removed parts of the training sets to explore the generalization behavior of our model, resulting in $|\mathcal{T}_1| = 450$ and $|\mathcal{T}_2| = 350$.

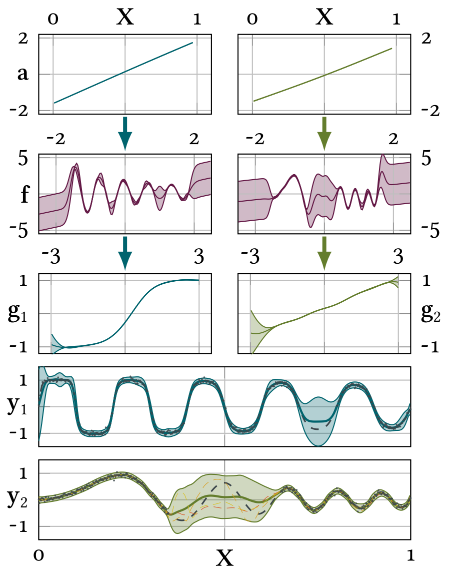
We use this setup to train our model using squared exponential kernels both in the conditionally independent GPs $\mathbf{a}_d$ and $\mathbf{g}_d$ and as smoothing kernels in $\mathbf{f}$. We can always choose one alignment and one warping to be the identity function to constrain the shared latent spaces $\mathbf{a}$ and $\mathbf{f}$ and provide a reference the other alignments and warpings will be relative to. Since we assume our artificial data simulates a physical system, we apply the prior knowledge that the alignment and warping processes have slower



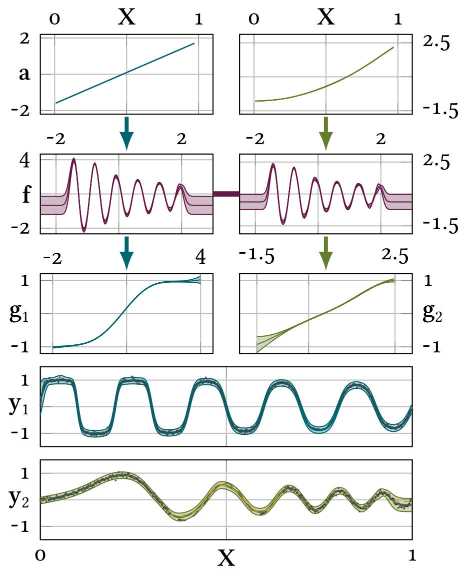
(a) Shallow GP with RBF kernel.



(b) Multi-Output GP with dependent RBF kernel.



(c) Deep GP with RBF kernels.



(d) AMO-GP with (dependent) RBF kernels.

Figure 4.3: A comparison of the AMO-GP with other GP models. The plots show mean predictions and a shaded area of two standard deviations. If available, the ground truth is displayed as a dashed line. Additional lines are noiseless samples drawn from the model. The shallow and deep GPs in Figures 4.3a and 4.3c model the data independently and revert back to the prior in y_2. Because of the nonlinear alignment, a multi-output GP cannot model the data in Figure 4.3b. The AMO-GP in Figure 4.3d recovers the alignment and warping and shares information between the two outputs.

dynamics compared to the shared latent function that should capture most of the observed dynamics. To this end we applied priors to the $\mathbf{a}_d$ and $\mathbf{g}_d$ that prefer longer length scales and smaller variances compared to $\mathbf{f}$. Otherwise, the model could easily get stuck in local minima like choosing the upper two layers to be identity functions and model the time-series independently in the $\mathbf{g}_d$. Additionally, our assumption of identity mean functions prevents pathological cases in which the complete model collapses to a constant function.

Figure 4.3d shows the AMO-GP's recovered function decomposition and joint predictions. The model successfully recovered a shared latent dampened sine function, a sigmoid warping for the first time-series, and an approximate quadratic alignment function for the second time-series. In Figures 4.3a to 4.3c, we show the training results of a standard GP, a multi-output GP and a three-layer deep GP on the same data. For all of these models, we used RBF kernels and, in the case of the deep GP, applied priors similar to our model to avoid pathological cases. In Table 4.1 we report test log-likelihoods for the presented models, which illustrate the qualitative differences between the models. Because all models are non-parametric and converge well, repeating the experiments with different initializations leads to very similar likelihoods.

Both the standard GP and deep GP cannot learn dependencies between time-series and revert to the prior where no data is available. The deep GP has learned that two layers are enough to model the data, and the resulting model is essentially a Bayesian warped GP that has identified the sigmoid warping for $\mathcal{T}_1$. Uncertainties in the deep GP are placed in the middle layer areas where no data are available for the respective time-series, as sharing information between the two outputs is impossible. In contrast to the other two models, the multi-output GP can and must share information between the two time-series. As discussed in Section 4.1 however, it is constrained to constant time-offsets and cannot model the nonlinear alignment in the data. Because of this, the model cannot recover the latent sine function and can only model one of the two outputs.

Pairs of Wind-Turbines

This experiment is based on real data recorded from a pair of neighboring wind-turbines in a wind-farm. The two time-series $\mathcal{T}_1$ and $\mathcal{T}_2$ shown in gray in Figure 4.5 record the respective power generation of the two turbines over the course of one and a half hours, which was smoothed slightly using a rolling average over 60 seconds. There are 5400 data points for the first turbine (blue) and 4622 data points for the second turbine (green). We removed two intervals (drawn as dashed lines) from the second

Table 4.1: Test-log-likelihoods for the models presented in Section 4.4.

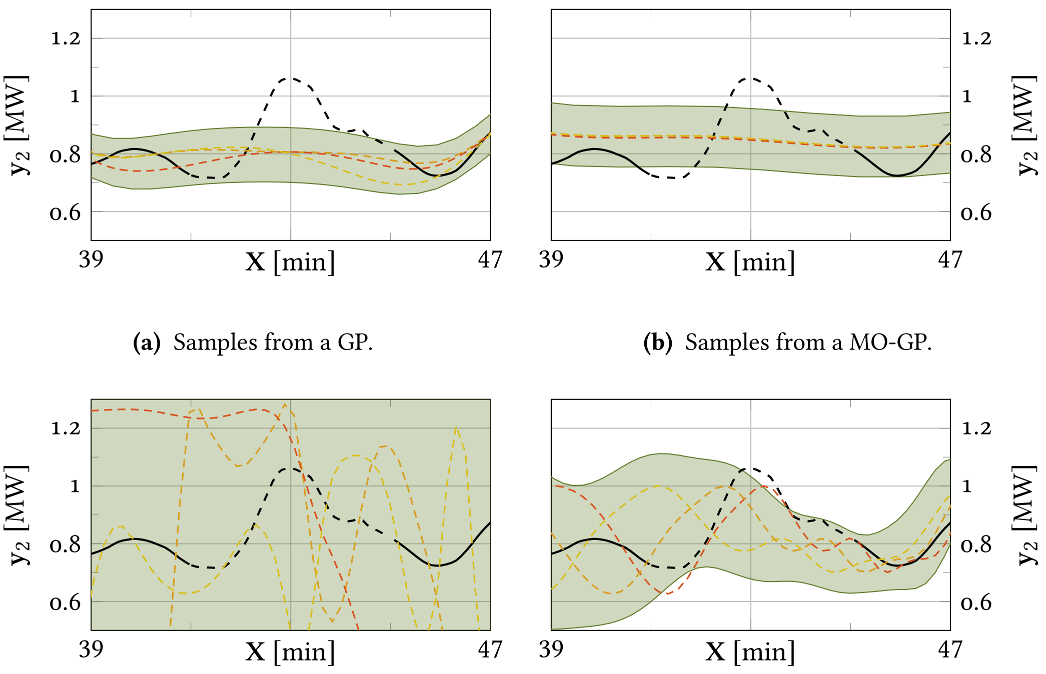
Experiment	Test set	GP	MO-GP	DGP	AMO-GP (Ours)
Artificial	$[0.7, 0.8] \subseteq \mathscr{T}_1$	-0.12	-0.053	0.025	**1.54**
	$[0.35, 0.65] \subseteq \mathscr{T}_2$	-0.19	-5.66	-0.30	**0.72**
Wind	$[40, 45] \subseteq \mathscr{T}_2$	-4.42	-2.31	-1.80	**-1.43**
	$[65, 75] \subseteq \mathscr{T}_2$	-7.26	-0.73	-1.93	**-0.69**

turbine's data set to inspect the behavior of the model with missing data. This allows us to evaluate and compare the generative properties of our model in Figure 4.4.

The power generated by a wind-turbine is mainly dependent on the speed of the wind fronts interacting with the turbine. For system identification tasks concerned with the behavior of multiple wind-turbines, associating the observations on different turbines due to the same wind fronts is an important task. However, it is usually not possible to directly measure these correspondences or wind propagation speeds between turbines, which means that there is no ground truth available. An additional problem is that the shared latent wind conditions are superimposed by turbine-specific local turbulence. Since these local effects are of comparable amplitude to short-term changes of wind speed, it is challenging to decide which parts of the signal to explain away as noise and which part to identify as the underlying shared process.

Our goal is the simultaneous learning of the uncertain alignment in time $\mathbf{a}$ and of the shared latent wind condition $\mathbf{f}$. Modeling the turbine-specific parts of the signals is not the objective, so they need to be explained by the Gaussian noise term. We use a squared exponential kernel as a prior for the alignment functions $\mathbf{a}_d$ and as smoothening kernels in $\mathbf{f}$. For the given data set, we can assume the output warpings $\mathbf{g}_d$ to be linear functions because there is only one dimension, the power generation, which in this data set is of similar shape for both turbines. Again we encode a preference for alignments with slow dynamics with a prior on the length scales of $\mathbf{a}_d$. As the signal has turbine-specific autoregressive components, plausible alignments are not unique. To constrain the AMO-GP, we want it to prefer alignments close to the identity function that we chose as a prior mean function.

In non-hierarchical models, this can be achieved by placing a prior on the kernel's variance, preferring smaller σ_a^2. In our case, the posterior space of the alignment process $\mathbf{a}$ is a latent space on which we have not placed any prior. The model can choose the posterior distribution of $\mathbf{u}_a$ in a way to counteract the constrained scale of



(a) Samples from a GP.

(b) Samples from a MO-GP.

(c) Samples from a DGP.

(d) Samples from the AMO-GP.

Figure 4.4: A comparison of noiseless samples drawn from a GP, a MO-GP, a DGP and the AMO-GP. The separation of uncertainties implied by the model structure of AMP-GP gives rise to an informative model. Since the uncertainty in the generative process is mainly placed in the relative alignment shown in Figure 4.5, all samples in Figure 4.4d resemble the underlying data in structure.

$\mathbf{K_{au}}$ and $\mathbf{K_{uu}^{-1}}$ and thereby circumvent the prior. To prevent this, we also place a prior on the mean of $\mathbf{u}_a$ to remove this degree of freedom.

Figure 4.5 shows the joint model learned from the data in which a_1 is chosen to be the identity function. The identified alignments match the physical conditions of the wind-farm. For the given turbines, time offsets of up to six minutes are plausible, and for most wind conditions, the offset is expected to be close to zero. For areas where the alignment is quite certain, however, the two time-series are explained with comparable detail. The model can recover unambiguous associations well and successfully places high uncertainty on the alignment in areas where multiple explanations are plausible due to the noisy signal.

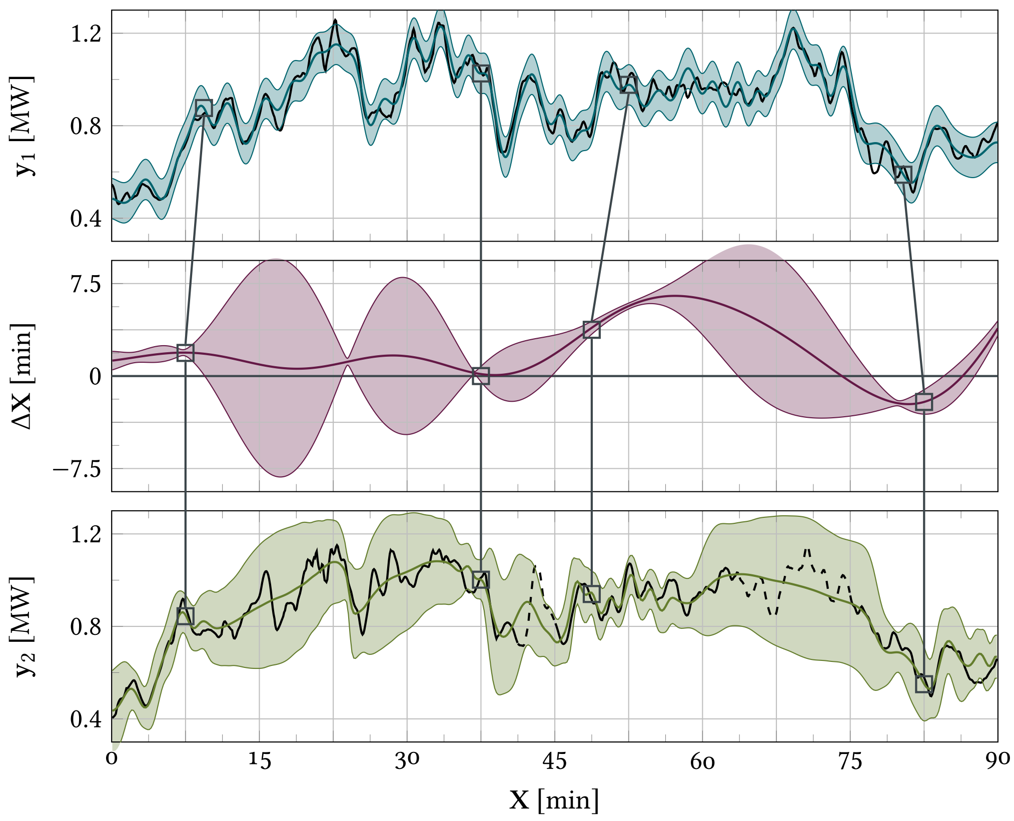
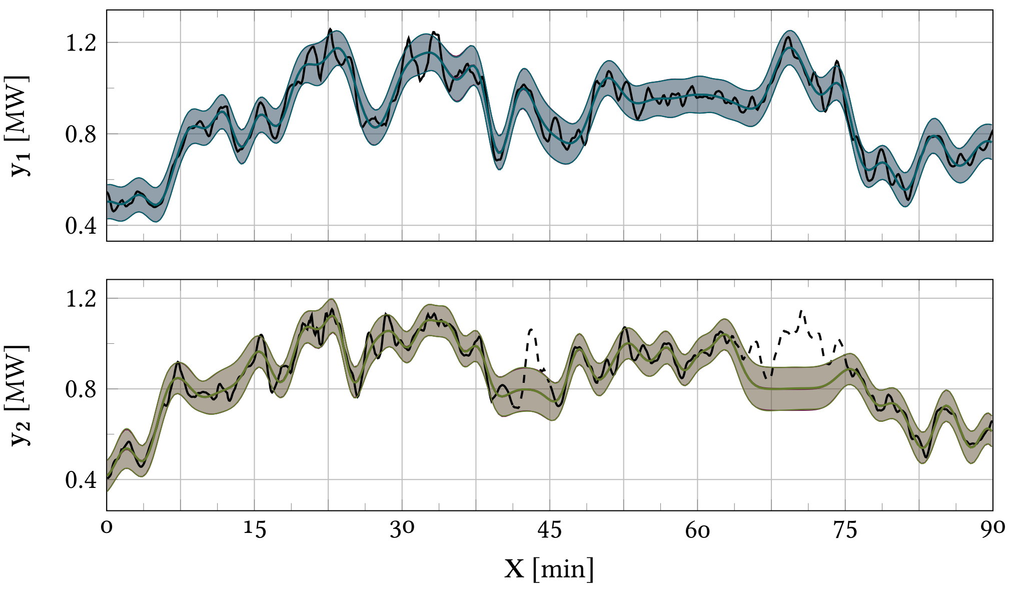


Figure 4.5: The joint posterior for two time-series $\mathbf{y}_1$ and $\mathbf{y}_2$ of power production for a pair of wind-turbines. The top and bottom plots show the two observed time-series with training data and dashed missing data. The AMO-GP recovers an uncertain relative alignment of the two time-series shown in the middle plot. Note that the sign chance in the alignment signifies a change in prevailing wind conditions. High uncertainty about the alignment is placed in areas where multiple explanations are plausible due to the high noise or missing data.

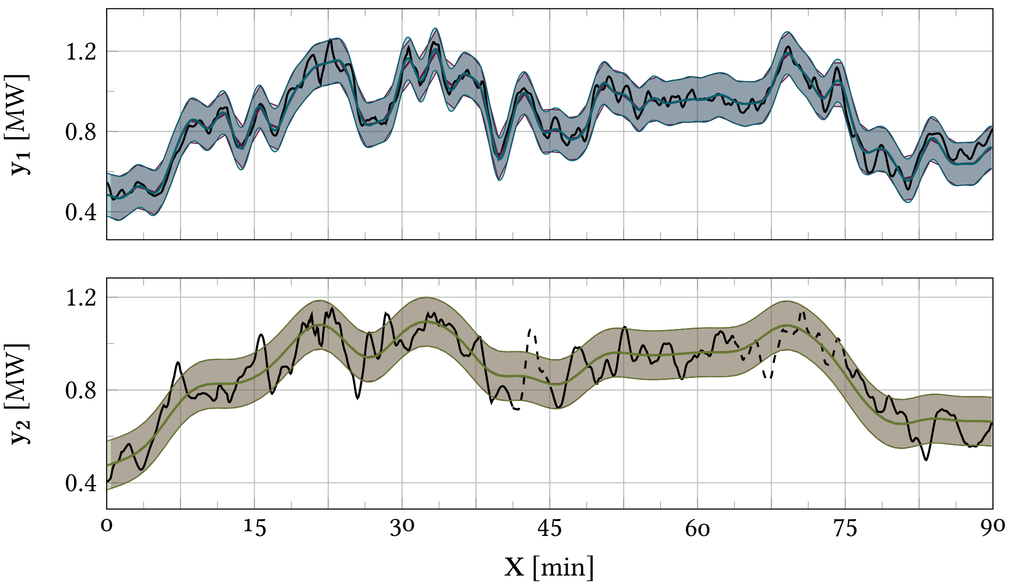
As expected, the uncertainty about the alignment also grows where data for the second time-series is missing. This uncertainty is propagated through the shared function and results in higher predictive variances for the second time-series. Because of the factorization in the model, however, we can recover the uncertainties about the alignment and the shared latent function separately. Figure 4.4 compares samples drawn from our model with samples drawn from a GP, a MO-GP and a DGP. The GP reverts to their respective priors when data is missing, while the MO-GP does not handle short-term dynamics and smoothens the signal enough such that the nonlinear alignment can be approximated as constant. Samples drawn from a DGP model showcase the complexity of a DGP prior. Unconstrained composite GPs are hard to reason about and make the model very flexible in terms of representable functions. Since the model's evidence is very broad, the posterior is uninformed, and inference is hard. Additionally, as discussed in Section 4.2 and [57], the nested variational compression bound tends to loosen with high uncertainties. AMO-GP shows richer structure: Due to the constraints imposed by the model, more robust inference leads to a more informed model. Samples show that it has learned that a maximum which is missing in the training data has to exist somewhere, but the uncertainty about the correct alignment due to the local turbulence means that different samples place the maximum at different locations in $\mathbf{X}$-direction.

In Figures 4.6 and 4.7, we compare joint predictions and the variational approximations for different model types in the wind example. Similar to the artificial data set, we show a standard GP in Figure 4.6a, a multi-output GP in Figure 4.6b, a deep GP in Figure 4.7a and AMO-GP in Figure 4.7b. All models were trained until convergence, and multiple runs result in very similar models. For all models, we used RBF kernels or dependent RBF kernels where applicable. Each plot shows the data in gray and two mean predictions and uncertainty bands. The first violet uncertainty band is the result of the variational approximation of the respective model. The second green or blue posterior is obtained via sampling. For both the GP and MO-GP, we used the SVGP approximation, and since the models are shallow, the approximation is almost exact.

Figure 4.7a showcases the difficulty of training a deep GP model and the shortcomings of the nested variational compression. The violet variational approximation is used for training and approximates the data comparatively well. As discussed above, the deep GP cannot share information, so the test sets cannot be predicted. The approximation tends to underestimate uncertainties when propagating them through the different layers, and because of this, uncertainties obtained via sampling vary considerably more. Only the variational approximation is considered for model selection. Because the deep GP is highly underspecified, the variational bound is very loose. The posterior obtained via ancestral sampling through the deep GP is considerably different and uninformative

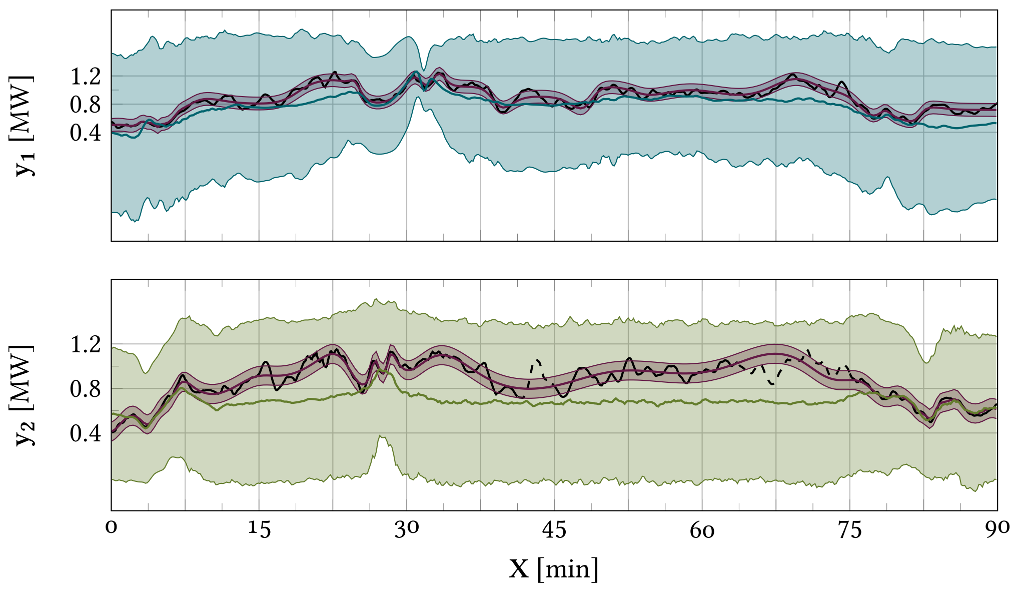


(a) GP

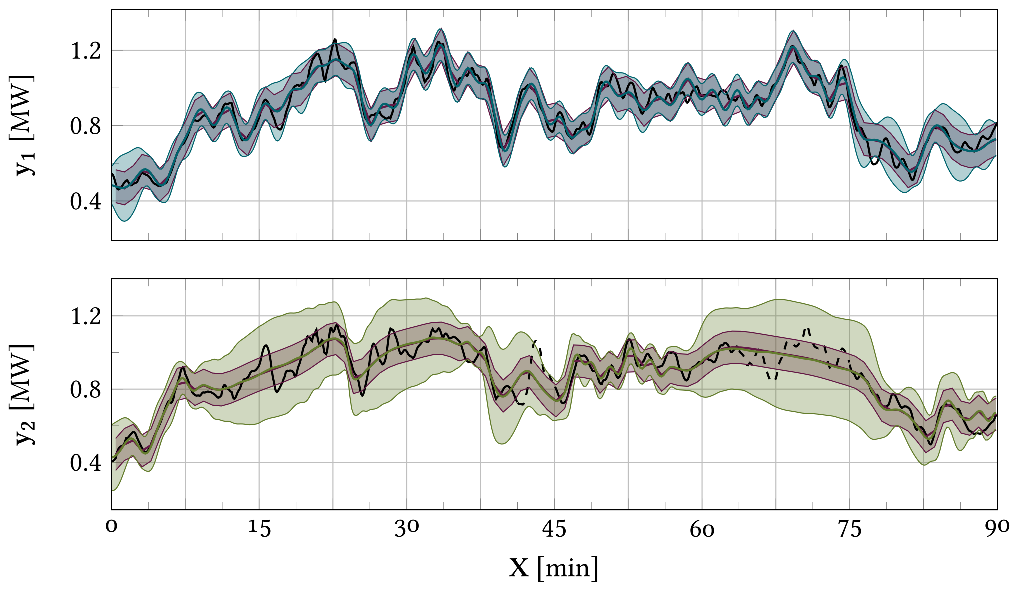


(b) MO-GP

Figure 4.6: Comparison of joint predictions (blue and green) and the variational approximations (violet). Both the GP and MO-GP are shallow models and the variational approximations are tight.



(a) DGP



(b) AMO-GP (Ours)

Figure 4.7: Comparison of joint predictions (blue and green) and the variational approximations (violet). The DGP model has few constraints and the variational bound is very loose. The knowledge encoded in AMO-GP leads to a more informed and thighter approximation.

as a result. Our approach has the same problem as the deep GP in principle in that the variational bound could become very loose. However, the structure of the hierarchy imposed through an interpretable generative process ensures that the different layers in the AMO-GP model distinct parts of the learning problem. As a result, the model avoids degenerate solutions, and the variational approximation is closer to the true posterior. The posteriors tend to disagree more in places when there is high uncertainty about the alignment.

4.5 Discussion

We have proposed the warped and aligned multi-output Gaussian process (AMO-GP), in which MO-GPs are embedded in a hierarchy to find shared structure in latent spaces. We extended convolution processes [18] with conditionally independent Gaussian processes on both the input and output sides, giving rise to a highly structured deep GP model. This structure can be used to both regularize the model and encode expert knowledge about specific parts of the system. By applying nested variational compression [57] to inference in these models, we presented a variational lower bound which combines Bayesian treatment of all parts of the model with scalability via stochastic optimization.

We compared the model with GPs, deep GPs and multi-output GPs on an artificial data set and showed how the richer model-structure allows the AMO-GP to pick up on latent structure which the other approaches cannot model. We then applied the AMO-GP to real-world data of two wind-turbines and used the proposed hierarchy to model wind propagation in a wind-farm and recover information about the latent non-homogeneous wind field. With uncertainties decomposed along the hierarchy, our approach handles ambiguities introduced by the stochasticity of the wind in a principled manner. This indicates the AMO-GP is a good approach for these kinds of dynamical systems, where multiple misaligned sensors measure the same latent effect.

We have shown how limitations in the independence assumptions of hierarchical GP approximations in Chapter 3 can be overcome by introducing dependence in the generative process. By modeling this dependence through a multi-output GP, we can derive an explicit posterior belief about how multimodal data interact. We have shown in Figure 4.4 that this belief leads to a rich structure in the samples drawn from a hierarchical model. It allows us to formulate priors that specify which aspects of a dynamical system should be inferred. However, we also observe that the structure of samples drawn from the model is only considered indirectly during training through the marginal likelihood. Similarly, the concrete shape of different model components is only influenced through the prior, and a semantically correct model cannot be recognized. Similar to how a correct data association is subjective, the correct alignment of time-series is subjective and subject to the problem domain.

In the next chapter, we explore how this subjectivity can be incorporated into the model itself. First, we will formulate a hierarchical structure for a problem in which model components can be immediately linked to system behavior and thus be interpreted by domain experts. Secondly, we not only seek to capture the generative process underlying a dataset but also incorporate the task the model will solve into model selection. Taking a reinforcement learning problem as an example, we show that semantic hierarchical structure increases data efficiency and allows domain experts to influence agent behavior through detailed insights into the dynamics of a system.

Chapter 5

Interpretable Reinforcement Learning

In Chapter 4, we formulated a hierarchically structured Gaussian process model to uncover latent information shared between multimodal time-series. We showed how this dependence both removes ambiguities in a learning problem and leads to more informative posteriors. While comparisons to less semantic generative models showed similar behavior on training data, we showed that a more informed model shows more desirable generalization behaviors. In a regression problem, the evaluation of desirable behavior is usually limited to the inspection of marginal distributions at test points and manual inspection of samples. In this chapter, we explore how we can formalize desirable model behavior through the task a model will be used to solve. We formulate a hierarchical model based on the DAGP model in Chapter 3 that is informed by abstract knowledge about a dynamic system and evaluate model behavior through a reinforcement learning task. We show that while misspecified models yield similar marginal distributions as discussed in the previous chapter, a well-specified model can solve the posed task significantly better.

Machine learning methods are designed to solve tasks where the underlying system we want to model is only partly known or understood. The hope when using machine learning methods is that we can reduce this uncertainty by exploiting statistical patterns in data generated from the underlying system. Reinforcement learning (RL) [111] is a machine learning paradigm designed to learn in a dynamic environment where we can specify a goal or have a notion of what a desirable behavior is. The goal of RL is to learn a policy that dynamically chooses actions in the environment to achieve a goal or behavior. Specifically, an RL agent's task is to learn a policy π that, given the current state $\mathbf{s}$ of an environment, chooses an action $\mathbf{a}$ to achieve the goal specified by a reward function r mapping system states to numerical rewards. To learn a policy, an RL system needs to understand the underlying dynamics governing the system, how the transition between states is affected by the actions taken. The next state $\mathbf{s}'$ is determined by the latent and possibly stochastic transition function $\mathbf{s}' = f(\mathbf{s}, \mathbf{a})$.

How the dynamical system is treated is one of the main distinctions among different approaches to RL. In model-based RL, the dynamics model is an explicit part of the system, while in the model-free counterpart, the transition dynamics are implicit and cannot be disentangled from the system.

Applying RL in an industrial setting often implies trying to derive an alternative, more efficient controller for an already existing system. In a critical application, it is unlikely that we will be able to deploy an untested policy, as this can lead to safety issues. In practice, we are often limited by previously collected data created by a different, possibly known, control mechanism in order to learn our model. In the literature, this scenario is referred to as batch RL [72], where we are presented with a set of state transitions $\mathscr{D} = \{(\mathbf{s}_n, \mathbf{a}_n, \mathbf{s}'_n)\}_{n=1}^{N}$ and are unable to interact with the original system to find a policy. In order to be able to derive an efficient policy in this scenario, we need to use the available data as efficiently as possible. Data efficiency in machine learning comes from reducing the search space of solutions. In other words, data efficiency arises from being able to exploit as much prior knowledge of the system as possible [97].

To be able to use the available data as efficiently as possible, we therefore need a model that provides explicit and interpretable handles such that we can introduce priors easily. The model-based approach to RL describes each component of the system in a modular fashion, thereby providing an interface to incorporate prior knowledge. The challenge is how these priors should be specified, and how they should be included, such that the hypothesis space can be limited in a manner coherent with our knowledge of the system.

Gaussian processes (GPs) are stochastic processes that can be used to specify probability distributions over the space of functions. While a GP specifies a distribution with support for all functions, it efficiently concentrates its probability mass to functions with specific characteristics. These characteristics make GPs well suited for RL as they do not impose hard constraints while still placing a significant structure on the space of functions. In [36, 37], the authors propose a model-based RL method where Gaussian processes are used as priors for the dynamics. They provide a principled approach for taking model uncertainty into account when evaluating the performance of a policy, thereby reducing the impact of model-bias. However, the approach has several restrictions: Transition dynamics are modeled as standard Gaussian processes (GPs), and policies and rewards must be of specific forms.

In this chapter, we show how we can alleviate some of the limitations of [36] to provide a richer and more efficient RL model. We show how we can introduce additional constraints on the dynamic model allowing for multiple transitional signatures to be active simultaneously. Incorporating this knowledge facilitates learning by allowing us to more precisely state what we want to learn, thereby significantly reducing the data requirements. Furthermore, decomposing the transition model into several parts allows us to use reward shaping [111] in order to discourage policies based on dynamic characteristics.

Introducing constraints on the dynamic model based on abstract knowledge is an inherently problem-dependent process. This work explores this process for the heteroscedastic and bimodal Wet-Chicken benchmark [52, 116] that is both easy to understand and challenging to model. A central challenge in this benchmark is how to formulate a model which can represent bimodalities. One approach is presented in [12], where multimodal regression tasks are interpreted as a density estimation problem. A high number of candidate distributions are reweighed to match the observed data without modeling the underlying generative process. Reformulated in a Bayesian framework using latent variables, this approach has been applied to the Wet-Chicken benchmark in [40, 41]. However, such models are hard to interpret as they do not yield explicit models for the different modalities or their relative importance. In this work, we are interested in formulating a dynamics model which yields new interpretable insights about the underlying system. We formulate a probabilistic model which contains such explicit models by interpreting the Wet-Chicken benchmark as a data association problem [8, 30]. While many probabilistic interpretations of this problem assume that the relative importance of different modes is constant [15, 75], we base our formulation on the DAGP model from Chapter 3 which learns a non-parametric model of each mode and where the associations between the modes themselves is further controlled by a non-stationary stochastic process.

After introducing the Wet-Chicken benchmark, we show how high-level knowledge about this system can be used to impose a Bayesian structure. We derive an efficient inference scheme for both the dynamics model and for probabilistic policy search based on variational inference. We show that this approach yields interpretable models and policies and is significantly more data-efficient than less interpretable alternatives.

5.1 The Wet-Chicken Benchmark

In the (two-dimensional) Wet-Chicken problem [52, 116], a canoeist is paddling in a two-dimensional river. The canoeist's position at time t is given by $\mathbf{s}_t = (x_t, y_t) \in \mathbb{R}^2$, where x_t denotes the position along the river and y_t the position across it. The river is bounded by its length $l = 5$ and width $w = 5$. There is a waterfall at the end of the

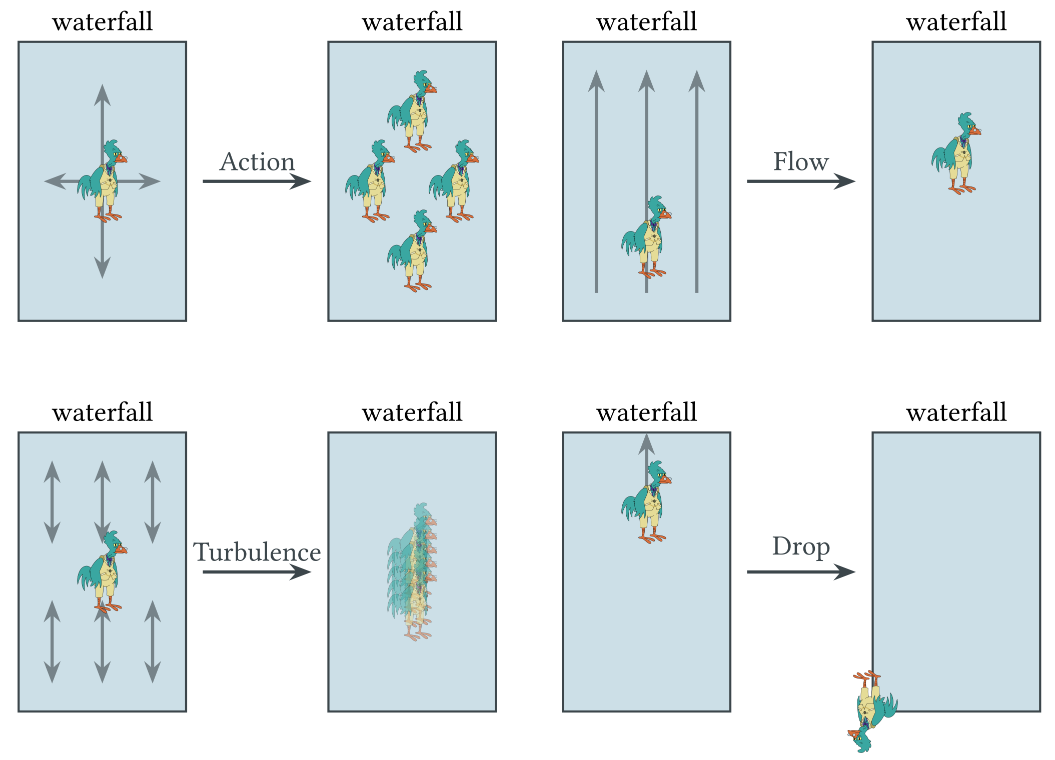


Figure 5.1: The core components of the 2D wet-chicken benchmark. An agent moves around a river with the goal to get as close to the waterfall as possible and remain there. At every time step, the river flows forward non-uniformly and also has a turbulent component. After falling, the agent restarts at maximum distance to the waterfall.

river at $x = l$. The canoeist wants to get close to the waterfall to maximize the reward $r(\mathbf{s}_t) = r_t = x_t$. However, if the canoeist falls down the waterfall, they have to start over at the initial position $(0, 0)$.

The river's flow consists of a deterministic velocity $v_t = 3 \cdot y_t w^{-1}$ and stochastic turbulence $b_t = 3.5 - v_t$, both of which depend on the position on the y-axis. The higher y_t is, the faster the river flows, but the less turbulent it becomes. The canoeist chooses his paddle direction and intensity via an action $\mathbf{a}_t = (a_{t,x}, a_{t,y}) \in [-1, 1]^2$. The transition function $f : (\mathbf{s}_t, \mathbf{a}_t) \mapsto \mathbf{s}_{t+1} = (x_{t+1}, y_{t+1})$ is given by

$$x_{t+1} = \begin{cases} 0 & \text{if } \hat{x}_{t+1} > l \\ 0 & \text{if } \hat{x}_{t+1} < 0 \\ \hat{x}_{t+1} & \text{otherwise} \end{cases} \qquad y_{t+1} = \begin{cases} 0 & \text{if } \hat{x}_{t+1} > l \text{ or } \hat{y}_{t+1} < 0 \\ w & \text{if } \hat{y}_{t+1} > w \\ \hat{y}_{t+1} & \text{otherwise} \end{cases} \tag{5.1}$$

where

$$\begin{aligned} \hat{x}_{t+1} &= x_t + (1.5 \cdot a_{t,x} - 0.5) + v_t + b_t \cdot \tau_t, \\ \hat{y}_{t+1} &= y_t + a_{t,y}, \end{aligned} \tag{5.2}$$

and $\tau_t \sim \mathbb{U}(-1, 1)$ is a uniform random variable that represents the turbulence. Figure 5.1 visualizes the core components of the transition function.

There is almost no turbulence at $y = w$, but the velocity is too high to paddle back. Similarly, the velocity is zero at $y = 0$, but the canoeist can fall down the waterfall unpredictably due to the high turbulence. A successful canoeist must find a balance between handling the stochasticity and velocities within the capabilities of the canoeist to get as close to the waterfall as possible. However, as the canoeist moves closer to the waterfall, the distribution over the next states become increasingly more bi-modal as the probability of falling increases. Together with the heteroscedasticity introduced by the turbulence dependent on the current position, these properties make the Wet-Chicken problem especially difficult for model-based reinforcement learning problems.

5.2 Probabilistic Policy Search

We are interested in finding a policy specified by the parameters $\boldsymbol{\theta}_\pi$ that maximizes the discounted return $J^\pi(\boldsymbol{\theta}_\pi) = \sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t) = \sum_{t=0}^{T} \gamma^t r_t$ with a constant discount factor $\gamma \in [0, 1]$. Starting from an initial state $\mathbf{s}_0$ we generate a trajectory of states $\mathbf{s}_0, \dots, \mathbf{s}_T$ obtained by applying the action $\mathbf{a}_t = \pi(\mathbf{s}_t)$ at every time step t. The next state is generated using the (latent) transition function f, yielding $\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$.

Many environments have stochastic elements, such as the random drift in the Wet-Chicken benchmark from Section 5.1. We take this stochasticity into account by interpreting the problem from a Bayesian perspective where the discounted return specifies a generative model whose graphical model is shown in Figure 5.2. Because

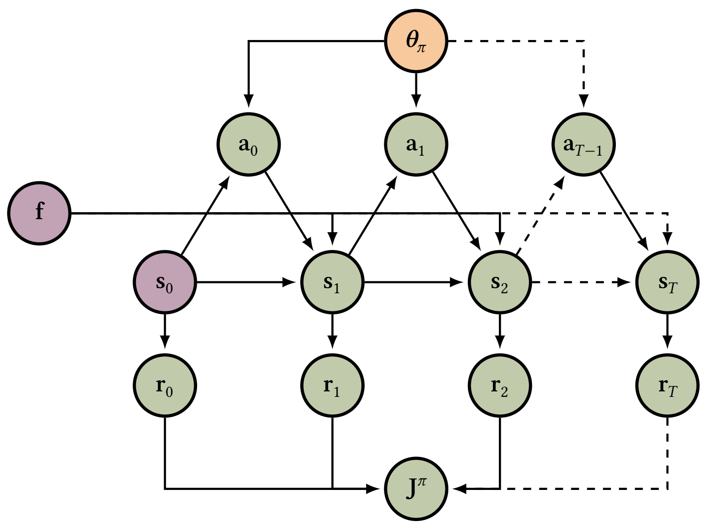


Figure 5.2: The generative process for the return J^π. It shows how starting from $\mathbf{s}_0$, a trajectory of length T is generated with the policy parameterized by $\boldsymbol{\theta}_\pi$. The return is generated by the rewards which depend on their respective states only.

of the Markov property assumed in RL, conditional independence between the states yields a recursive definition of the state probabilities given by

$$\begin{aligned} \mathrm{p}(\mathbf{s}_{t+1} \,|\, f, \boldsymbol{\theta}_\pi) &= \int \mathrm{p}(f(\mathbf{s}_t, \mathbf{a}_t) \,|\, \mathbf{s}_t, \mathbf{a}_t)\, \mathrm{p}(\mathbf{a}_t \,|\, \mathbf{s}_t, \boldsymbol{\theta}_\pi)\, \mathrm{p}(\mathbf{s}_t)\, \mathrm{d}\mathbf{a}_t\, \mathrm{d}\mathbf{s}_t, \\ \mathrm{p}(r_t \,|\, \boldsymbol{\theta}_\pi) &= \int \mathrm{p}(r(\mathbf{s}_t) \,|\, \mathbf{s}_t)\, \mathrm{p}(\mathbf{s}_t \,|\, \boldsymbol{\theta}_\pi)\, \mathrm{d}\mathbf{s}_t. \end{aligned} \tag{5.3}$$

With stochasticity or an uncertain transition model, the discounted return becomes uncertain and the goal can be reformulated to optimize the expected return

$$\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)] = \sum_{t=0}^{T} \gamma^t\, \mathbb{E}_{\mathrm{p}(\mathbf{s}_t|\boldsymbol{\theta}_\pi)}[r_t]. \tag{5.4}$$

A model-based policy search method consists of two key parts [36]. First, a dynamics model is learned from the state transition data. Second, this dynamics model is used to learn the parameters $\boldsymbol{\theta}_\pi$ of the policy π which maximize the expected return $\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)]$. We discuss both steps in the following.

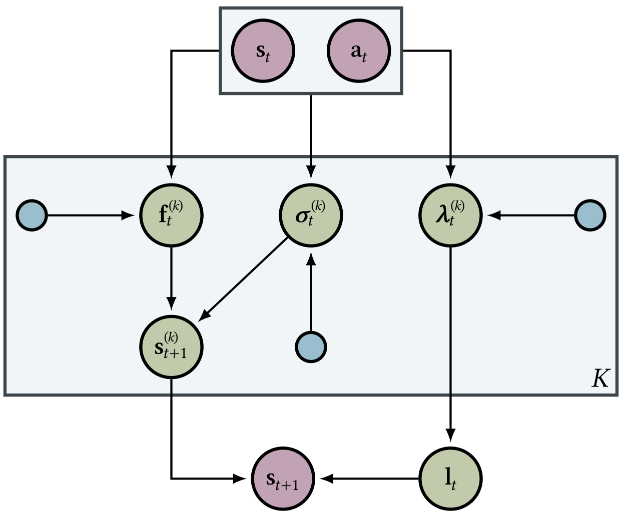


Figure 5.3: The graphical model for the DAGP-based transition model. This model separates the flow-behavior of the river $\mathbf{f}_t$, the heteroscedastic noise process $\boldsymbol{\sigma}_t$ and the possibility of falling down $\boldsymbol{\lambda}_t$. Latent variables $\mathbf{l}_t$ represent the belief that the t^{th} data point is a drop event.

An Interpretable Transition Model

We formulate a probabilistic transition model based on high-level knowledge about the Wet-Chicken benchmark. Importantly, we do not formulate a specific parametric dynamics model as would be required to derive a controller. Instead, we make assumptions on a level typically available from domain experts.

We encode that given a pair of current state and action $\hat{\mathbf{s}}_t = (\mathbf{s}_t, \mathbf{a}_t)$, the next state $\mathbf{s}_{t+1}$ is generated via the combination of three things: the deterministic flow-behavior of the river $\mathbf{f}_t$, some heteroscedastic noise process $\boldsymbol{\sigma}_t$ and the possibility of falling down $\boldsymbol{\lambda}_t$. This prior imposes structure which allows us to explicitly state what we want to learn from the data and where we do not assume prior knowledge: How does the river flow? What kind of turbulence exists? When does the canoeist fall? How do the actions influence the system?

Each question is explicitly answered by one of the model's components. In Section 5.3, we will visualize these components and discuss how they can be used by experts to gain new insights about the system. Additionally, interpretable transition models help to build trust in derived policies: Since experts can assess the plausibility of the transition model, successful policies are unlikely to behave unexpectedly on the true system.

We formulate a graphical model in Figure 5.3 using the DAGP model from Chapter 3, which allows us to handle the multi-modality introduced by falling down the waterfall and extend it to handle the heteroscedastic properties of the problem. We specify this separation via the marginal likelihood

$$\mathrm{p}(\mathbf{s}_{t+1} \,|\, \hat{\mathbf{s}}_t) = \int \mathrm{p}(\mathbf{s}_{t+1} \,|\, \boldsymbol{\sigma}_t, \mathbf{f}_t, \mathbf{l}_t)\, \mathrm{p}(\mathbf{l}_t \,|\, \hat{\mathbf{s}}_t)\, \mathrm{p}(\boldsymbol{\sigma}_t \,|\, \hat{\mathbf{s}}_t)\, \mathrm{p}(\mathbf{f}_t \,|\, \hat{\mathbf{s}}_t)\, \mathrm{d}\boldsymbol{\sigma}_t\, \mathrm{d}\mathbf{l}_t\, \mathrm{d}\mathbf{f}_t, \tag{5.5}$$

where $\mathbf{f}_t = \left(\mathbf{f}_t^{(1)}, \dots, \mathbf{f}_t^{(K)}\right)$. The marginal likelihood consists of the two GPs $\mathrm{p}(\boldsymbol{\sigma}_t \,|\, \hat{\mathbf{s}}_t)$ and $\mathrm{p}(\mathbf{f}_t \,|\, \hat{\mathbf{s}}_t)$ and the two likelihoods

$$\begin{aligned} \mathrm{p}(\mathbf{s}_{t+1} \,|\, \boldsymbol{\sigma}_t, \mathbf{f}_t, \mathbf{l}_t) &= \prod_{k=1}^{K} \mathcal{N}\left(\mathbf{s}_{t+1} \,\middle|\, \mathbf{f}_t^{(k)}, \left(\boldsymbol{\sigma}_t^{(k)}\right)^2\right)^{\mathbb{I}(l_t^{(k)}=1)}, \\ \mathrm{p}(\mathbf{l}_t \,|\, \hat{\mathbf{s}}_t) &= \int \mathcal{M}(\mathbf{l}_t \,|\, \operatorname{softmax}(\boldsymbol{\lambda}_t))\, \mathrm{p}(\boldsymbol{\lambda}_t \,|\, \hat{\mathbf{s}}_t)\, \mathrm{d}\boldsymbol{\lambda}_t \end{aligned} \tag{5.6}$$

where $\mathcal{M}$ denotes a multinomial distribution. These likelihoods describe the regression and the classification tasks implied by the problem respectively: In our case, we use $K = 2$ modes, one for staying in the river and one for falling down the waterfall. For every data point we infer a posterior belief $\mathrm{p}(\mathbf{l}_t)$ about which mode the data point belongs to, as we assume this separation can not be predetermined using expert knowledge.

We place independent GP priors on the $\mathbf{f}^{(k)}$, $\boldsymbol{\sigma}^{(k)}$ and $\boldsymbol{\lambda}^{(k)}$. Given the data, a fixed set of assignments $\mathbf{L}$, our modeling assumptions imply independence between the K modes. However, this independence is lost if the assignments are unknown, and a discrete optimization problem has to be solved for joint inference over the different modes and the association problem. We approximate the exact posterior via a factorized variational distribution

$$\mathrm{q}(\mathbf{f}, \boldsymbol{\lambda}, \boldsymbol{\sigma}, \mathbf{U}) = \prod_{k=1}^{K} \prod_{t=1}^{T} \mathrm{q}(\mathbf{f}_t^{(k)}, \mathbf{u}^{(k)})\, \mathrm{q}(\boldsymbol{\lambda}_t^{(k)}, \mathbf{u}_\lambda)^{(k)}\, \mathrm{q}(\boldsymbol{\sigma}_t^{(k)}, \mathbf{u}_\sigma)^{(k)} \tag{5.7}$$

which introduces variational inducing inputs and outputs $\mathbf{U}$ as described in Chapter 2. These inducing inputs independently characterize the respective model parts and enable us to do inference via stochastic optimization.

The variational parameters are optimized by minimizing a lower bound on the marginal likelihood which can be efficiently computed via sampling and enables stochastic optimization:

$$
\begin{aligned}
\mathscr{L}_{\text{DAGP}} &= \mathbb{E}_{\mathrm{q}(\mathbf{F},\boldsymbol{\lambda},\boldsymbol{\sigma},\mathbf{U})}\left[\log \frac{\mathrm{p}\left(\mathbf{S}', \mathbf{F}, \boldsymbol{\lambda}, \boldsymbol{\sigma}, \mathbf{U} \middle| \hat{\mathbf{S}}\right)}{\mathrm{q}(\mathbf{F}, \boldsymbol{\lambda}, \boldsymbol{\sigma}, \mathbf{U})}\right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{\mathrm{q}(\mathbf{f}_t)}[\log \mathrm{p}(\mathbf{s}'_t \,|\, \mathbf{f}_t, \boldsymbol{\lambda}_t, \boldsymbol{\sigma}_t)] + \sum_{t=1}^{T} \mathbb{E}_{\mathrm{q}(\boldsymbol{\lambda}_t)}[\log \mathrm{p}(\mathbf{l}_t \,|\, \boldsymbol{\lambda}_t)] \\
&\quad - \sum_{k=1}^{K} \mathrm{KL}(\mathrm{q}\left(\mathbf{u}^{(k)}, \mathbf{u}_\lambda^{(k)}, \mathbf{u}_\sigma\right)^{(k)} \,\|\, \mathrm{p}\left(\mathbf{u}^{(k)}, \mathbf{u}_\lambda^{(k)}, \mathbf{u}_\sigma\right)^{(k)})
\end{aligned}
\tag{5.8}
$$

We obtain an explicit representation of the GP posteriors during variational inference, which allows us to efficiently propagate samples through the model to simulate trajectories used for policy search. Predictions for an unknown state $\hat{\mathbf{s}}_*$ are mixtures of K independent Gaussians given by,

$$
\begin{aligned}
\mathrm{q}(\mathbf{s}'_* \,|\, \hat{\mathbf{s}}_*) &= \int \sum_{k=1}^{K} \mathrm{q}(l_*^{(k)} \,|\, \hat{\mathbf{s}}_*)\, \mathrm{q}(\mathbf{s}_*^{\prime(k)} \,|\, \hat{\mathbf{s}}_*) \,\mathrm{d}\mathbf{l}_* \\
&= \int \sum_{k=1}^{K} \mathrm{q}(l_*^{(k)} \,|\, \hat{\mathbf{s}}_*)\, \mathrm{q}(\mathbf{s}_*^{\prime(k)} \,|\, \mathbf{f}_*^{(k)} \boldsymbol{\sigma}_*^{(k)})\, \mathrm{q}(\mathbf{f}_*^{(k)}, \boldsymbol{\sigma}_*^{(k)} \,|\, \hat{\mathbf{s}}_*) \,\mathrm{d}\mathbf{l}_* \,\mathrm{d}\mathbf{f}_* \,\mathrm{d}\boldsymbol{\sigma}_* \\
&\approx \sum_{k=1}^{K} \tilde{l}_*^{(k)} \tilde{\mathbf{s}}_*^{\prime(k)}.
\end{aligned}
\tag{5.9}
$$

We sample from the assignment process $\mathbf{l}_*$ and heteroscedastic noise process $\boldsymbol{\sigma}_*$. The K predictive posteriors $\mathrm{q}(\mathbf{s}_*^{\prime(k)} \,|\, \mathbf{f}_*^{(k)} \boldsymbol{\sigma}_*^{(k)})$ are then given by K independent shallow GPs and can be computed analytically.

Policy Learning

After training a transition model, we use the variational posterior $\mathrm{q}(\mathbf{s}_{t+1} \,|\, \hat{\mathbf{s}}_t)$ to train a policy by sampling roll-outs and optimizing policy parameters via stochastic gradient descent on the expected return $\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)]$. The expected return is approximated using the variational posterior given by

$$\mathbb{E}[J^\pi(\boldsymbol{\theta}_\pi)] = \sum_{t=0}^{T} \gamma^t \, \mathbb{E}_{\mathrm{p}(\mathbf{s}_t|\boldsymbol{\theta}_\pi)}[\mathbf{r}_t] \approx \sum_{t=0}^{T} \gamma^t \, \mathbb{E}_{\mathrm{q}(\mathbf{s}_t|\boldsymbol{\theta}_\pi)}[\mathbf{r}_t] \tag{5.10}$$

$$= \int \sum_{t=0}^{T} \left[\gamma^t \, \mathbb{E}_{\mathrm{q}(\mathbf{s}_t|\boldsymbol{\theta}_\pi)}[\mathbf{r}_t] \right] \mathrm{p}(\mathbf{s}_0) \prod_{t=0}^{T-1} \mathrm{q}(\mathbf{s}_{t+1} \,|\, \mathbf{s}_t, \boldsymbol{\theta}_\pi) \, \mathrm{d}\mathbf{s}_0 \dots \mathrm{d}\mathbf{s}_T \tag{5.11}$$

$$\approx \frac{1}{P} \sum_{p=1}^{P} \sum_{t=0}^{T} \gamma^t r_t^p . \tag{5.12}$$

We expand the expectation to explicitly show the marginalization of the states in the trajectory. Due to the Markovian property of the transition dynamics, the integral factorizes along t. The integral is approximated by averaging over P samples propagated through the model starting from a known distribution of initial states $\mathrm{p}(\mathbf{s}_0)$. State transitions can be efficiently sampled from the variational posterior of the dynamics model by repeatedly taking independent samples of the different GPs.

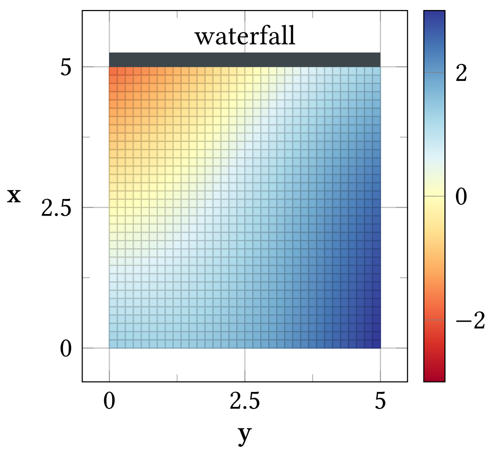
The expected return in (5.10) can be optimized using stochastic gradient descent via the gradients

$$\nabla_{\theta_\pi} J^\pi(\theta_\pi) \approx \frac{1}{P} \sum_{p=1}^{P} \sum_{t=0}^{T} \gamma^t \nabla_{\theta_\pi} r_t^p \tag{5.13}$$

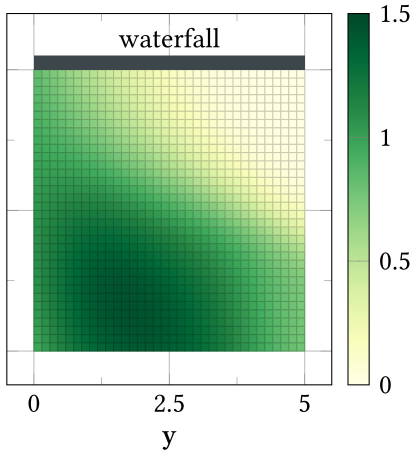
of the Monte Carlo approximation as they are an unbiased estimator of the true gradient. The gradients of the samples can be obtained using automatic differentiation tools such as TensorFlow [1]. The P roll-outs can be trivially parallelized. Importantly, we only need a small number of Monte Carlo samples at every iteration, since we use the gradients of the samples directly.

5.3 Experiments

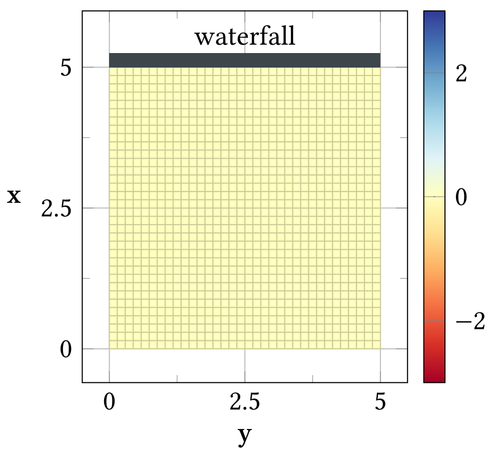
To solve the Wet-Chicken problem, we first train the dynamics model on batch data sampled from the true dynamics and then optimize neural policies with respect to this dynamics model. As the DAGP-based dynamics model is designed to be interpretable, we first discuss how, additionally to a joint posterior, the independent posteriors of its components yield insights about the Wet-Chicken problem. We then show how successful policies can be found with fewer observations compared to the model-free NFQ [89] and model-based Bayesian Neural Networks with latent variables (BNN+LV) [41], two



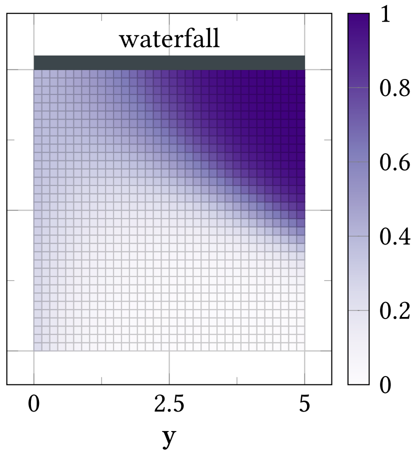
(a) Flow $\Delta\mathbf{x}^{(1)}$



(b) Heteroscedastic turbulence $\boldsymbol{\sigma}_x$



(c) Drop $\mathbf{x}'^{(2)}$



(d) Drop probability $\boldsymbol{\lambda}^{(2)}$

Figure 5.4: The separation of different aspects of the Wet-Chicken in DAGP-based transition models benchmark yields new and interpretable information about the underlying dynamics. The different parts of the model explicitly show flow speeds (Figure 5.4a), turbulence (Figure 5.4b), drop behavior (Figure 5.4c) and drop probabilities (Figure 5.4d) with respect to the current position in the river and action $a = 0$. The model has learned that the river is turbulent on the left and fast on the right, leading to consistent medium drop probabilities on the left due to stochasticity and a sharp boundary on the right, where the flow speed dominates. Note that a drop resets the position to the initial state irrespective of the current state. It is therefore correctly learned to be represented by the constant zero function (Figure 5.4c).

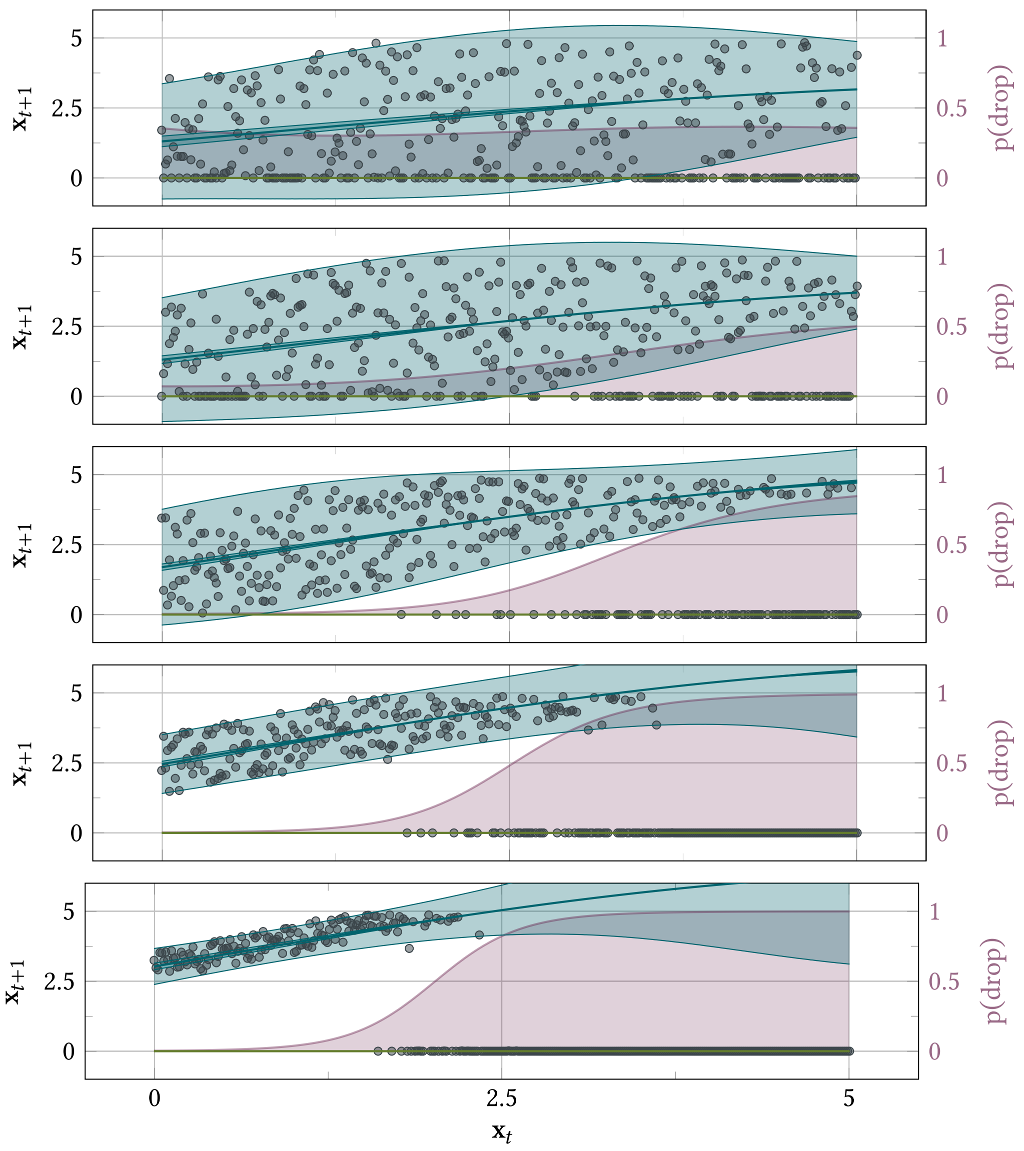


Figure 5.5: Linear cuts through the DAGP-based transition model with the waterfall on the right at $x_t = 5$. The plots show the dependency between x_t and x_{t+1} with respect to the action $a_t = 0$ and, from top to bottom, $y_t \in \{0, 1, 2.5, 4, 5\}$. The DAGP-based transition model successfully separates the two modes introduced through flow (blue) and drop (green) behaviors and predicts the probability of being assigned to the drop mode (violet). While drops can be modeled using a constant noiseless function, the flow speed (gradient and bias) and heteroscedastic noise (variance) varies in the different cuts. For low y_t, the river flows slowly but is very turbulent, while for high y_t, the river flows fast but deterministically.

approaches which do not make use of high-level expert knowledge. Thirdly, we show how the human-interpretable components of the dynamics models can be used for reward shaping, allowing us to formulate a requirement for conservative policies.

Dynamics Model

The benchmark has two-dimensional state and action spaces from which we sample uniform random transitions with varying N in the range 100 to 5000. For $N \geq 250$, our model can identify the underlying dynamics. In Figure 5.5 we show the joint predictive posterior of a DAGP-based transition model. The different plots show linear cuts through the Wet-Chicken system with respect to the action $a_t = 0$ and $y_t \in \{0, 1, 2.5, 4, 5\}$. The transition model has successfully identified the two modes introduced through flow and drop behaviors and their relative importance. These cuts require an additional examination to recover new knowledge about the system's behavior. In contrast, the separation of the learning problem in the DAGP-based dynamics model gives us explicit and separate posteriors about the different system components via the independent GP posteriors shown in Figure 5.4. This belief can directly be reasoned about with experts to evaluate the environment in which policies will be trained, raising confidence in their correctness.

While drops can be modeled using a constant noiseless function, the flow speed and heteroscedastic turbulence vary throughout the system. For low y, the river flows slowly but is very turbulent, while for high y, the river flows fast but deterministically. In the turbulent regime, falling down is possible but not certain for most x, while in the flow dominated regime, a drop becomes highly probable under a certain distance from the waterfall. Note that even though the turbulence as defined in Section 5.1 is independent of x, the heteroscedastic noise process has uncovered the implicit dependency for high x as most possible turbulence values lead to falling down and thus assignment to the other mode. Similarly, the flow speed shown in Figure 5.4a is negative in the top left corner, since the flow mode models the position after one step under the condition of not falling. As most turbulence into the direction of the waterfall leads to a drop, the posterior mean is further away from the waterfall as the turbulence dominates the low flow speed on the left side of the river.

Policy Learning

Given a posterior for the dynamics model, we now train a neural policy using probabilistic model rollouts. We sample initial states from the training data, use a horizon of

Table 5.1: Comparison of expected returns Using interpretable DAGP-based transition models with structurally informative priors, successful policies can be learned based on 250 observations. In contrast, about 2500 observations are needed to find a policy using the model-free NFQ. As GP based transition models are not capable of representing bimodal dynamics, training does not yield successful policies.

N	NFQ	BNN+LV	GP	DAGP
100	0.66 ± 0.16	—	**1.41 ± 0.01**	1.18 ± 0.09
250	1.71 ± 0.07	1.62 ± 0.20	1.54 ± 0.01	**2.33 ± 0.01**
500	1.60 ± 0.10	2.18 ± 0.07	1.56 ± 0.01	**2.25 ± 0.01**
1000	1.99 ± 0.06	2.27 ± 0.01	2.13 ± 0.01	**2.32 ± 0.01**
2500	2.26 ± 0.02	**2.30 ± 0.01**	1.91 ± 0.01	2.28 ± 0.01
5000	**2.33 ± 0.01**	2.30 ± 0.01	1.91 ± 0.01	2.28 ± 0.01

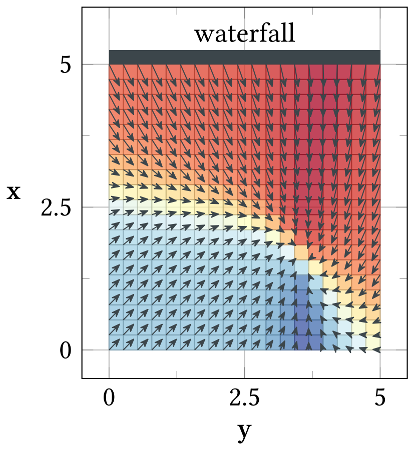


Figure 5.6: A successful Wet-chicken policy. It has found a trade-off between the unpredictability on the left and the uncontrollable speed on the right. The policy approaches a sweet-spot on the right side and tries to stay in place.

Table 5.2: As the different components of a DAGP-based transition model are easily interpretable, they can be used for reward shaping. We formulate a conservative reward function r_{cons} which penalizes drops and can easily be evaluated in the transition model. A conservative policy has a lower return but avoids the waterfall more often.

Training Reward	Original Return	Conservative Return	Drop %
r_{orig}	**2.32 ± 0.01**	−1.22 ± 0.02	21.8 ± 0.2
r_{cons}	2.17 ± 0.01	**−1.00 ± 0.01**	**18.9 ± 0.1**

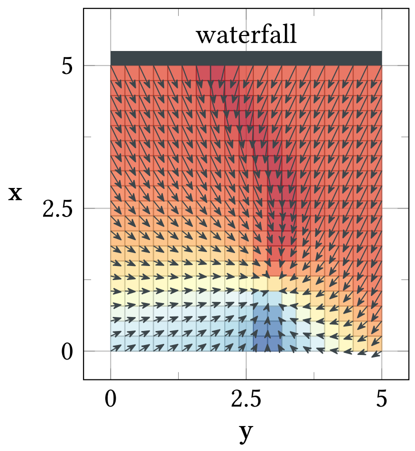


Figure 5.7: A conservative Wet-Chicken policy. By cautiously approaching a sweet-spot further away from the waterfall, it avoids the waterfall more often than a standard policy.

$T = 5$ steps and average over $P = 20$ samples with $\gamma = 0.9$. We use a two-layer neural network, with 20 ReLU-activated units each, as our policy parametrization. For every state transition, we sample independently from the different model components to generate a sample for the next state using (5.9). This incorporates both the stochasticity in the system introduced via heteroscedastic noise and the Bayesian uncertainty about the correct model in the policy search. During training, the policy thus implicitly learns to consider the stochasticity of the Wet-Chicken benchmark as different sample-trajectories generate gradients with respect to different realizations of the stochasticity of the Wet-Chicken benchmark. Figure 5.6 shows how a successful policy has found a trade-off between the unpredictability on the left and the uncontrollable speed on the right.

In Table 5.1, we compare policy search based on the DAGP-based dynamics model with a standard GP dynamics model and NFQ. We present expected returns for training runs with different amounts of data averaged over ten experiments together with standard errors. A policy applying uniformly random actions yields a return of about 1.5, and a return above 2.2 indicates that a successful policy has been found. We ran NFQ for 20 full model learning and sampling iterations using a neural network with one 10-unit hidden layer with sigmoid activations.

A standard GP cannot model heteroscedastic noise or multi-modality. For any point in the input space, the GP can, therefore, only predict that the agent will always fall down, never fall down, or, via very high uncertainties, that any state in the system is possible. None of these possibilities represent the dynamics well enough to allow the policy search to derive a policy, illustrating our need for a more structured model. For $N \geq 250$, the DAGP-based dynamics model identifies the underlying dynamics well, and policies can be found reliably.

BNN+LV is a more expressive model that can represent both heteroscedasticity and multi-modality. Due to the model's structure, however, it is hard to incorporate high-level expert knowledge, and therefore, more structure has to be learned from the data. BNN+LV reliably finds good policies for $N \geq 1000$ and sometimes finds good policies for $N = 500$. As this approach is model-based and formulates a reasonable general-purpose prior on the dynamics, the results fall between the more informed DAGP, which is successful with less data, and NFQ, which is more uninformed.

NFQ is a model-free approach. Instead of learning a dynamics model and using rollouts in that model to find a good policy, NFQ directly models the optimal Q-function and thus the optimal policy. A Q-function represents the expected return after taking a specified action in a specified state. Since the expected return already takes into account both the heteroscedasticity and multi-modality of the system, the Q-function itself can be modeled with a standard function approximator, such as a neural network. Thus, no special modeling is needed when applying NFQ to the Wet-Chicken benchmark and, given enough data, NFQ can find successful policies. However, at the same time, not modeling the dynamics explicitly also prevents us from utilizing the high-level expert knowledge we have about the system, thus increasing the required amount of data: Using DAGP-based dynamics models, a successful policy can be found with about an order of magnitude fewer observations.

Reward Shaping

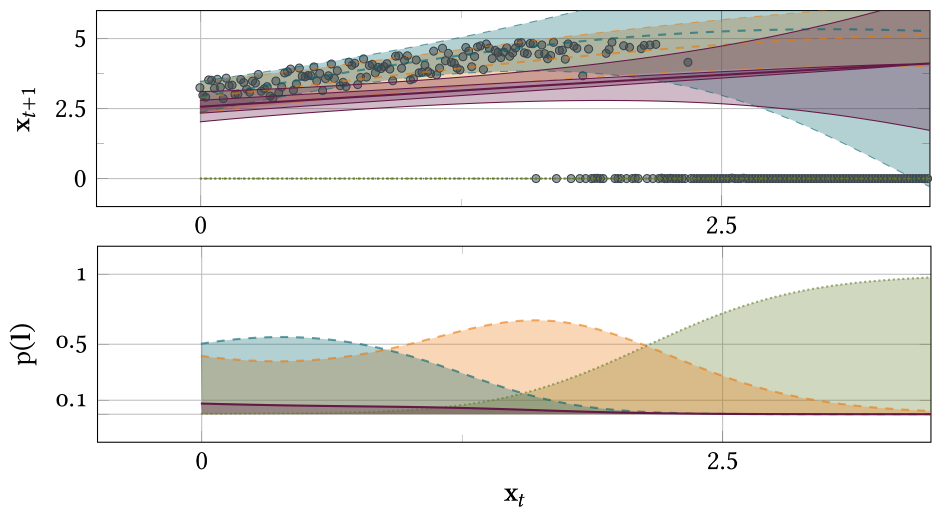
We have shown how a dynamics model informed by high-level expert knowledge increases data efficiency. A further advantage of the decomposition of the dynamics model in interpretable components is that the predictions of these components can be used to influence the policy search. In this example, we want to find a more conservative policy which, when compared to Figure 5.6, sacrifices some return in order to avoid falling down the waterfall.

Any successful agent has the implicit incentive to avoid drops as they move the canoeist away from the waterfall. However, a successful policy still accepts that it will fall sometimes due to turbulence. To encourage more conservative behavior, we use a conservative reward

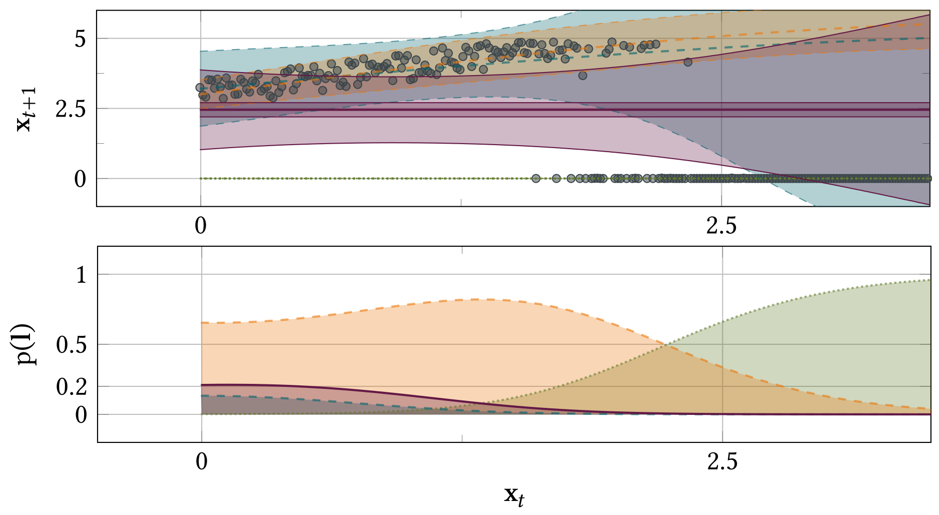
$$r_{\text{cons}}(\mathbf{s}) = r_{\text{orig}}(\mathbf{s}) \cdot (1 - \mathrm{p}(\text{drop} \,|\, \mathbf{s})) - 5 \cdot \mathrm{p}(\text{drop} \,|\, \mathbf{s}) \tag{5.14}$$

which includes the original Wet-Chicken reward function $r_{\text{orig}}((x, y)) = x$. For every state, the DAGP-based dynamics model yields an explicit drop-probability which can easily be evaluated. The conservative reward punishes a high drop probability reweighed with the maximum original reward $\max_{\mathbf{s}} r_{\text{orig}}(\mathbf{s}) = 5$.

Figure 5.7 shows a resulting conservative policy. Such a policy avoids both the turbulent states on the left and the fast-flowing states on the right. It tries to reach a sweet spot, which, compared to Figure 5.6, is further away from the waterfall and therefore safer. We compare 10 runs with $N = 1000$ observations using the original reward and the conservative reward in Table 5.2. The resulting conservative policies yield a lower return than the more aggressive default policies but reliably reduce drop probabilities as well. The interpretable nature of the dynamics models has allowed us to influence policy behaviors.



(a) Transition model with $K = 4$ and successful policy training



(b) Transition model with $K = 4$ and failed policy training

Figure 5.8: Comparison of linear cuts through two DAGP-based transition models with $K = 4$ at $y_t = 5$ and $a_t = 0$. The respective upper plots show the predictive posterior of the different modes while the lower plots show assignment probabilities to the different modes. For both models, one mode (green, dotted) model drops and two modes (blue and yellow, dashed) represent flow behavior. A third more uninformed mode (violet) is almost irrelevant in the first model but explains some data through a high noise variance in the second model. A significant amount of predictions from the second model are uninformed, leading to the failure of the policy search.

Table 5.3: Comparison of expected returns for different settings of K. Well-specified models with $K = 2$ reliably solve the benchmark problem with low amounts of data. Mis-specified models show higher variance in their results and do not reliably solve the benchmark.

	DAGP				
N	$K = 1$	**K = 2**	$K = 3$	$K = 4$	$K = 5$
250	1.41 ± 0.01	$\mathbf{2.33 \pm 0.01}$	1.64 ± 0.38	1.31 ± 0.25	1.65 ± 0.08
500	1.54 ± 0.01	$\mathbf{2.25 \pm 0.01}$	1.97 ± 0.23	1.48 ± 0.21	2.14 ± 0.10
1000	2.13 ± 0.01	$\mathbf{2.32 \pm 0.01}$	1.99 ± 0.17	2.09 ± 0.12	2.16 ± 0.09
2500	1.91 ± 0.01	$\mathbf{2.28 \pm 0.01}$	2.15 ± 0.03	2.06 ± 0.12	2.17 ± 0.03
5000	1.91 ± 0.01	$\mathbf{2.28 \pm 0.01}$	2.19 ± 0.06	1.95 ± 0.16	2.08 ± 0.13

Effects of Model Misspecification

In Section 5.2 we have formulated a dynamics model informed by high-level expert knowledge. One important insight we assumed is the bimodal nature of the Wet-Chicken problem introduced by the waterfall. In Section 5.3, we compared our model to standard GPs and showed that modeling multi-modality is critical to solving Wet-Chicken. We extend this comparison in this experiment and discuss the effects of model misspecification on our model's performance. Specifically, we investigate the case where additional modes are available to the dynamics model to solve the underlying data association problem.

Table 5.3 shows results for $K \in \{1, \dots, 5\}$, where $K = 1$ is equivalent to standard GPs. All models have been trained for the same number of iterations, and, for $K > 1$, all models have comparable marginal likelihoods. While 250 data points are enough with $K = 2$ to reliably solve the Wet-Chicken problem, more data is needed until working policies can be found with $K > 2$ (for example for $K = 5$, double the data was required for one of the runs to find a working policy). Most notably, performance fluctuates significantly with misspecified models for different repetitions of the same experiment, and good policies can not be found reliably.

Using the additional modes available, the model can now find representations of the systems where multiple modes jointly represent the river's flow. This showcases how data association problems are inherently ill-posed in general [8, 30]. The additional representative power for $K > 2$ introduces symmetries in the optimization landscape that both complicate training [75, 78], and lead to undesired generalization behavior which is not driven by knowledge about the underlying system.

An example for undesired generalization is shown in Figure 5.8 which compares two models trained with $K = 4$ and $N = 2500$. While both models explain the overall training data well, the cuts through the system at $y_t = 5$ give an intuition why the first model leads to a successful policy, while the second model does not. Both models represent drops via one of the modes and share the remaining three modes to jointly explain the flow behavior. In the first model, two alternating modes have learned essentially equivalent models, and a third more uninformed mode is almost irrelevant. The second model is similar, but the uninformed mode's model is closer to the RBF prior and more relevant. Note that due to the high noise variance, this choice of the model still explains the data only slightly worse. Still, the second model does not generalize according to the underlying system. A significant amount of predictions from the second model are uninformed, leading to the failure of the policy search.

Significantly longer training or specialized optimization schemes may lead to robust inference for $K > 2$. However, this experiment shows the significance of encoding available abstract prior knowledge to avoid pathologic model behaviors. Models for $K = 2$ both reliably identify the system using standard optimization methods and yield immediately interpretable results.

5.4 Discussion

We have presented a Bayesian reinforcement learning model-based on non-parametric Gaussian process priors. The model is motivated by the observation that in real-world scenarios, high-level prior knowledge of the system dynamics is often available. We believe that many tasks are characterized by dynamics that can be decomposed into several attributes. For example, when a physical structure is excited by a force oscillating at its natural frequency, its response will change drastically. The approach we have presented is based on learning a modular dynamic model that decomposes this type of transitional behavior into separate components. The model learns both the individual components and the underlying structure of how the components interact within the system. The use of Gaussian process priors to quantify the uncertainty within components allows us to perform probabilistic policy search.

The interpretable structure of our model facilitates data-efficient learning by incorporating prior knowledge. We showed experimentally how an informed dynamics model reduces the data requirements compared to a model-free approach. Furthermore, the same knowledge can be used as a means for directing the policy search by discouraging solutions which exhibit a specific dynamic, such as avoiding falling down the waterfall in the Wet-Chicken benchmark.

The model presented in this chapter is derived from the DAGP model in Chapter 3 and therefore shares its limitations. Most importantly, the variational approximation which factorizes between layers cannot capture the full Bayesian posterior for ambiguities in the association problem. In Chapters 3 and 4, we showed that it is hard to distinguish qualitatively different solutions automatically. This difficulty is a consequence of the difficulty to formally describe desirable model behavior besides high marginal likelihoods on training or test data. In this chapter, we presented an approach to achieve this formal description by evaluating models on a reinforcement learning task they are designed to solve. A successful model both shows an expert-interpretable internal structure and good performance on downstream tasks that depend on a correct representation of the dynamical system.

Chapter 6

Discussion and Future Work

This thesis explored how to formulate Bayesian structured models with Gaussian process models. In Chapter 2, we gave an introduction to Bayesian machine learning based on statistical learning theory. A machine learning problem was characterized as an under-specified computational problem that requires additional subjective assumptions to solve. In a regression problem, finitely many observations are used to select an infinite-dimensional object, a function that explains them well. Bayesian structured models enabled us to formalize abstract assumptions about these functions while also allowing us to gain new insights from data. We have shown that due to their rich internal structure, judging the performance of structured models through the risk minimization algorithm may not be sufficient to identify the desirable models.

Chapter 3 discussed how Bayesian white-box models can be relaxed with Gaussian processes to allow data-driven insights and strong scalability. We introduced a Bayesian approach to the data association problem, where we consider a data set that has been generated by a mixture of processes, and where we are interested in factorizing the data into these components. A fully Bayesian model can produce such a factorization and yield models that explain observations well. However, since empirical risk minimization only evaluates the predictive distribution of a model and does not consider its internal structure, hierarchical models can introduce qualitative ambiguities: As long as they result in the same predictive distribution and are equally plausible under a structural prior, Bayesian model selection cannot distinguish between more and less interpretable or desirable solutions. In the general data association problem, formalizing the desirable solutions via some specific dependency structures is a hard problem from both the modeling and the inference points of view.

Chapter 4 considered how to achieve such a formalization in the more specific problem domain of time-series alignment. We started with a black-box deep Gaussian process model and added structure to constrain the model to yield interpretable results and trustworthy predictions. We viewed the power production of different wind-turbines in a wind-farm as a multi-modal time-series with shared latent information, the wind fronts interacting with the turbines. A hierarchical structure is introduced by the alignment problem of how wind fronts move through the system. By formulating a strong prior on the dependence between modes through a multi-output GP and encoding knowledge about the underlying physical system, ambiguities can be avoided, and rich structure can be learned. A strong prior simplifies inference but also limits what can be learned from data based on subjective notions of what makes a model correct.

Chapter 5 explored how this subjectiveness can be incorporated into the model itself to allow semantic model-selection. Taking the wet-chicken reinforcement learning problem as an example, a model is formulated based on high-level prior knowledge about the system dynamics decomposing the inference task into interpretable components. The correctness of the model can be evaluated through experts inspecting the interpretable components and the downstream performance in the reinforcement learning task. We showed that while misspecified models yielded similar predictive distributions to semantically correct models, a well-specified model can solve the posed task significantly better.

In this chapter, we explore further how taking downstream tasks into account can affect modeling decisions and discuss challenges during inference. A Bayesian posterior of a hierarchical generative process with general function approximators can be very complex, and the fewer constraints are put on it, the more heterogeneous it can be. The inference schemes for hierarchical Gaussian process models introduced in Section 2.7 rely on variational independence assumptions between components to achieve computational tractability. In Chapter 4, we saw that these inference schemes can struggle when posteriors become complex. In Section 6.1, we present an intuitive argument for why inference schemes based on factorizations between layers cannot represent heterogeneous posteriors. Empirical risk minimization through high marginal likelihoods in Bayesian model selection is not always enough to identify semantically correct hierarchical models as it does not consider internal structure. Undesired explanations of the data can be removed from the posterior by formulating priors with strong constraints, such as in Chapter 4. However, this approach requires extensive prior knowledge and limits what can be learned from data. In Section 6.2, we argue from a more general point of view and explore if it is always desirable for components of a structured model to explain the data. We show that models with suboptimal marginal likelihoods can perform well in hierarchical systems. Following the reasoning of including tasks in structured models further, we explore how tasks can be included

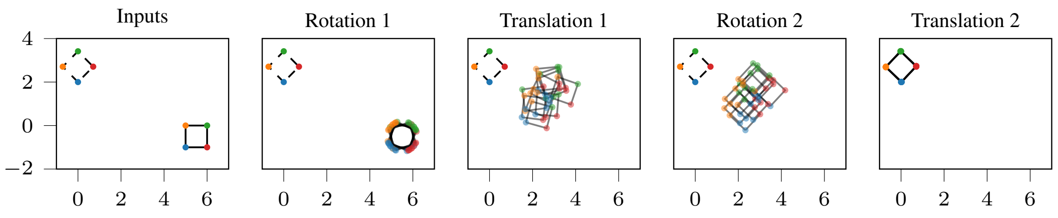


Figure 6.1: The rotated squares example. An overparameterized composite model should map the solid rectangle onto the dashed one. The required affine transformation is parameterized as $T_2 \circ R_2 \circ T_1 \circ R_1$, where R_i and T_i are rotations and translations. Different possible solutions show the complexity of the solution space. Approximating R_i and T_i as independent transformations does not allow us to capture a full posterior but collapses to a single solution.

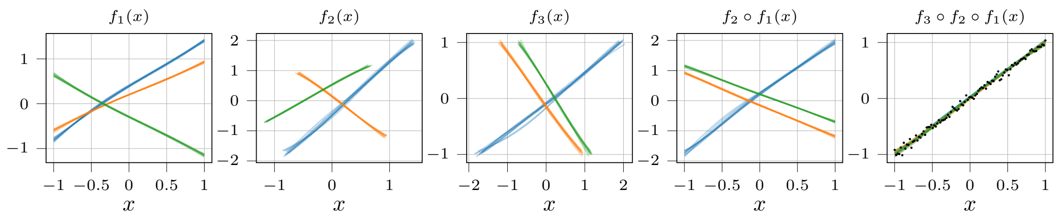


Figure 6.2: The identity function example. An identity function is parameterized as a composition of three linear functions with positive or negative slope. Different colors show three deep GP posteriors. Since the correct choice of the third layer depends on the first two layers, and independent approximation collapses to a single solution.

in a Bayesian inference problem directly in Section 6.3. In Section 6.4, we discuss possible further directions of research.

6.1 Inference in Hierarchical Models

The experiments in Chapter 3 showed that the data association problem generally does not have a unique solution. Besides exchanging modes entirely, the spatial priors in the Bayesian association model allow for qualitatively different explanations of the data, all of which conform similarly well to the prior and result in similar marginal likelihoods. Since all explanations are plausible, a full Bayesian posterior should contain all possible associations and generating processes. Marginalizing all plausible model realizations would take all possible solutions of the assignment problem into account. This full posterior is highly complex, however, and contains strong dependencies between the different components of the hierarchical model. In this section, we describe why the variational inference schemes from Section 2.7 based on strong independence assumptions can only represent posteriors of limited complexity. For additional details and detailed experiments, we refer to the associated publication [118].

Figure 6.1 illustrates this complexity on a toy example of mapping a square to another square in the two-dimensional euclidean plane. We formulate an overparameterized hierarchical model by representing the required affine transformation as the chain $T_2 \circ R_2 \circ T_1 \circ R_1$, with the T_i representing translations and the R_i representing rotations. The task can be solved by an arbitrary choice of T_1 and R_1 and corresponding choices $T_2 = T_* \circ T_1^{-1}$ and $R_2 = R_* \circ R_1^{-1}$ with $T_* \circ R_*$ representing the unique correct transformation from one square to the other. This example illustrates that the notion of a correct hierarchical posterior is problematic: A posterior model might assign a non-zero probability to all possible choices of the first rotation and transformation and a delta-distribution for the correct corresponding choice of the second rotation and transformation. Alternatively, a posterior might choose T_1 and R_1 to be identity mappings and $T_2 = T_*$ and $R_2 = R_*$. While the first posterior might be considered a more correct Bayesian posterior correctly representing unidentifiable parts of the learning problem, the second posterior is arguably simpler and represents the uniqueness of the correct solution better. Since both solutions explain the data equally well, this question might not be a problem in practice if the goal is to design a model that provides a high marginal likelihood of the data. In situations where reasoning about the internal structure of a hierarchical model such as in Chapter 4 or generalization behavior such as in Chapter 5 come into focus, informative posterior distributions over hierarchical generative processes can allow us to build more interpretable models and represent multiple qualitatively different solutions.

The original variational inference scheme for deep GP models by Damianou et al. [32] is prohibitively expensive for large data sets. To overcome this limitation, the inference schemes presented in Section 2.7 make strong independence assumptions between different components in a deep GP. While these assumptions allow training deep GPs with large amounts of data, assuming independence between the different f_i in a compositional prior $f_L \circ \ldots \circ f_1$ significantly limits the expressiveness of model posteriors, since every realization of the composition must map the same input to the same output. Figure 6.2 shows a composition of three linear functions, each of which has either a positive or a negative slope. Together, they need to represent the identity function. The choice of slope for f_3 depends on the choice of f_1 and f_2. The colors represent different results of training. Since the f_i are assumed to be independent, the only way to ensure that every realization of the posterior fits the data is to collapse to deterministic functions. This intuition also holds for general deep GPs, where qualitatively different choices in mappings closer to the input cannot lead to corresponding choices closer to the output.

The model discussed in Chapter 4 yields expressive and interpretable uncertainties by formulating a detailed and informed prior paired with a learning problem with inherently high uncertainties due to noise. Relaxing these requirements to less constrained models requires advances in inference schemes that can represent more expressive posteriors. Recent work extends the variational approximations with dependencies between layers [118] or proposes inference based on stochastic gradient Hamilton Monte Carlo [42, 53]. While in their current state, such inference schemes are computationally expensive, they present promising directions of research.

6.2 Surrogate Models For Bayesian Optimization

Above, we have argued that, in hierarchical models, a high marginal likelihood may not be sufficient to identify a desirable solution because the marginal likelihood does not take internal structure into account. In this section, we consider a complementary situation, where a model is embedded in a hierarchy. We consider Bayesian optimization (BO) [104], where an (often non-hierarchical) regression model is used to solve an optimization problem based on limited observations of the objective function. We make an argument why surrogate models with suboptimal marginal likelihood can be the model of choice for solving the downstream task. For additional details about the model and detailed experiments, we refer to the associated publication [16].

BO is a method for finding the optimum of functions that are unknown and expensive to evaluate. Let $f : \mathcal{X} \to \mathbb{R}$ be an unknown, noise-free objective function defined on a bounded subset $\mathcal{X} \subset \mathbb{R}^k$ for some $k \in \mathbb{N}$. The goal of BO is to solve the global optimization problem of finding

$$\mathbf{x}_* \in \operatorname*{arg\,min}_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}). \tag{6.1}$$

By fitting a surrogate model to samples of an unknown objective, the BO procedure iteratively picks new samples for evaluation that are believed to be the most informative about where the optimum is located. In real-world problems, the objective function is often not a well-behaved function, and a suitable model is difficult to specify. A misspecified model can be especially problematic in sequential decision making tasks such as BO, where the model is not only used to locate the optimum based on the collected data but also to decide where to collect data for future decisions.

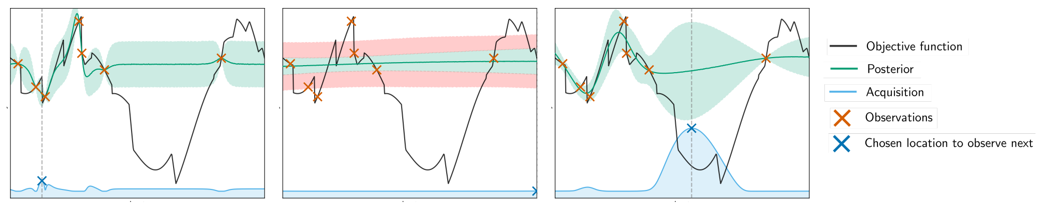


Figure 6.3: A comparison of Gaussian process posteriors for a noise-free GP (left), a homoscedastic GP with Gaussian noise (center) and a LGP (right). A noise-free GP learns a low lengthscale to explain the rapid oscillations in the data, while a homoscedastic GP explains them with a high noise level. Both surrogates lead to poorly informed decisions for the next BO query via the EI-acquisition. The LGP-surrogate sacrifices local exactness to recover global trends and yields an informed posterior.

There are many approaches to avoiding model misspecification in surrogate models for BO based on additional structure. For example, GP surrogates can be augmented with warpings to model non-stationarity [105], separate the input space via tree-structures [62] or optima can be searched via pairwise comparison [51]. While carefully chosen extensions can avoid misspecification, inference over more sophisticated surrogates often requires more observations, which is not desirable in the BO setting. Importantly, the ultimate goal of BO is to locate the optimum, not to model the objective as precisely as possible. In practice, surrogates with high complexity often perform worse compared to simpler ones even if the former contains the true objective function while the latter does not.

To formalize this observation, we consider the family of objective functions f that can be represented as a composition

$$f(\mathbf{x}) = g(\mathbf{x}, \mathbf{h}), \quad \mathbf{h} = h(\mathbf{x}), \tag{6.2}$$

where g is a well-behaved function that can be nicely modeled by a BO surrogate and the nuisance function $h(\mathbf{x})$ encodes the structures the surrogate model struggles to capture. Instead of building a complex surrogate model with minimal model misspecification, we propose to explicitly trade-off accuracy in modeling the objective with the efficiency of capturing informative structures from small amounts of data. For example, local oscillations and discontinuities are less important to capture in a BO setting than describing global trends.

Assuming additive structure $f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x})$ and choosing the nuisance function to be input-independent noise $h(\mathbf{x}) \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathrm{I})$ recovers GP regression with a Gaussian likelihood. For richer structure, we place a GP prior on $g(\mathbf{x}, \mathbf{h})$ and infer independent latent variables $\mathbf{h} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathrm{I})$ for all observations $\mathbf{x}$, giving rise to a Latent Gaussian process (LGP) [15, 85, 120, 124]. Intuitively, increasing the absolute value of $\mathbf{h}$ is penalized by the prior but allows the surrogate to not interpolate an observation exactly and to ignore local structure. Similar to using noiseless predictions with Gaussian likelihoods, we ignore the nuisance parameters $\mathbf{h}$ in the search for the optimum and focus on the global trends captured by $g(\mathbf{x}, \mathbf{0})$.

Figure 6.3 compares three surrogates on a function with challenging local structure. A standard noiseless GP interpolates all observations exactly and must capture all local variations. Since this surrogate quickly reverts to the prior between observations, it cannot capture global trends and therefore is not very helpful in solving the global optimization problem. The LGP-based surrogate ignore local structure and does not interpolate observations exactly, which leads to a reduced marginal likelihood. However, the model uncovers a global trend, which allows the BO-scheme to identify the correct area of interest.

The experiments in [16] give empirical evidence that this intuition holds for many BO tasks. The approach performs well in high-dimensional problems, where global trends tend to matter more in low-data regimes due to the curse of dimensionality. The idea that a surrogate in BO need not model the objective perfectly but yield informative insights for the optimization task was not formally included in the model formulation in this section. Instead, we considered a class of surrogates that implicitly have this property. In the next section, we go one step further and include the task of policy search in the inference problem of reinforcement learning explicitly.

6.3 Bayesian Reinforcement Learning

In statistical learning, the goal is to learn about an unknown latent probability distribution via a limited number of observations. The goal of learning is to recover some projection or functional of interest of the latent distribution. In this thesis, we have mostly focused on the regression setting, where the goal is to learn the marginal $\mathrm{p}(\mathbf{y} \,|\, \mathbf{x})$ of a data distribution $\mathrm{p}(\mathbf{x}, \mathbf{y})$. We have formulated hierarchical models to incorporate prior knowledge about the generative process that informs the data distribution. We have argued that a high marginal likelihood may not be enough to identify a desirable hierarchical model that, besides reproducing data, also captures the true generative process well.

In Chapter 5, we saw that well-specified models solve decision problems in reinforcement learning significantly better than misspecified problems, even if they represent

Table 6.1: Mapping of the statistical learning methodology to the reinforcement learning problem. An optimal policy is characterized by the Bellman principle. An RL algorithm must find a policy based on a limited amount of interaction with the system.

$$\begin{tikzcd}
p(\mathbf{z}) \arrow[r, "S"] \arrow[dr, "F"'] & \mathcal{S} \arrow[d, dashed, "A"] \\
 & \mathcal{F}
\end{tikzcd}$$

	Statistical learning	Reinforcement learning
$p(\mathbf{z})$	Latent distribution	True system-dynamics
$\mathcal{S}$	Observations	Batch or online data
$\mathcal{F}$	Functional of interest	Optimal value
$F : \mathcal{P}_{\mathcal{Z}} \to \mathcal{F}$	Functional operator	Bellman principle
$S : \mathcal{P}_{\mathcal{Z}} \to \mathcal{S}$	Sampling operator	Exploration
$A : \mathcal{S} \to \mathcal{F}$	Learning algorithm	Policy search

training data equally well. In this section, we formalize this insight and formulate the policy search problem in reinforcement learning via a generative process. This representation allows us to include the requirement for a good performance in the inference problem and reason about it in a Bayesian manner. In Table 6.1, we show how policy search in reinforcement learning problem can be formulated as a statistical learning problem. Based on latent true system dynamics, our goal is to identify the optimal value function and thus an optimal policy based on a limited number of interactions with the system. The optimal value is defined via the Bellman principle [111], which is hard to apply directly in general. Instead, we observe batch or online data and apply an algorithm for policy search.

Our goal is to formulate the policy search in reinforcement learning as an inference problem in a hierarchical model. We start by interpreting the definition of the T-step value function

$$J^{\pi}(\mathbf{s}_0) = \mathbb{E}\left[\sum_{t=1}^{T} \gamma^t r(\mathbf{s}_t) \,\middle|\, f, \pi\right] \tag{6.3}$$

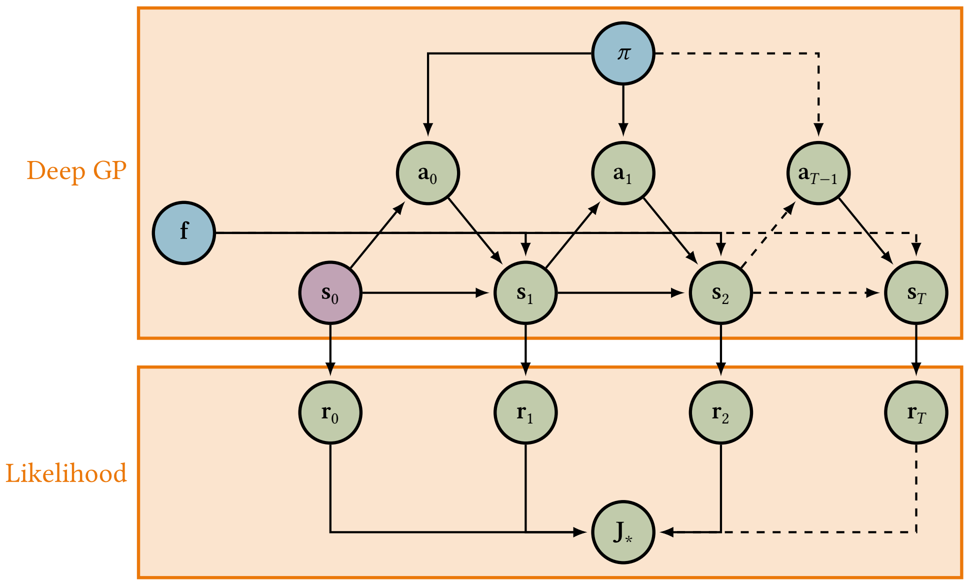


Figure 6.4: An interpration of the generative process of the optimal value as a hierarchical GP model. A transition model $\mathbf{f}$ and policy $\boldsymbol{\pi}$ parameterize the value through a deep GP and a special likelihood function.

as a generative process with the associated graphical model in Figure 6.4. Starting with the initial state $\mathbf{s}_0$, the trajectories are generated using the transition dynamics f and the policy π. If both the transition dynamics and the policy are represented as GPs, the graphical model of a trajectory is a recurrent deep Gaussian process. The states are mapped to rewards using the known reward function r and added to produce the value J^π, thereby defining a likelihood function. Instead of considering the value of any arbitrary policy π, we now reason about the uniquely defined optimal value J_*. Since the optimal value can be parameterized by an optimal policy π_*, it can be represented in our hierarchical model. Our goal is to formulate an equivalent to the marginal likelihood of the optimal value function and derive an inference scheme.

There are two distinct sources of uncertainty in this model. First, the knowledge of the dynamical system can be imperfect, or the system can be stochastic. Both effects make trajectories non-deterministic, even if the initial state and policy are fixed. This kind of uncertainty is reflected in the expected value of the original value definition, and we retain these uncertainties. Second, we do not maintain a candidate policy but approximate the optimal value instead. Since our knowledge about the optimal policy is imperfect, so is our knowledge about the optimal value. Both uncertainties are captured in the distribution $\mathrm{p}(J_* \mid \mathbf{s}_0)$.

To formulate an equivalent of the marginal likelihood, we can bound the optimal value function. We assume without loss of generality that the reward function is bounded and that $\max_{\mathbf{s}} r(\mathbf{s}) = 0$. Then we have that

$$\forall \mathbf{s} \forall \boldsymbol{\pi} \, : \, J^{\pi}(\mathbf{s}) \leq J^{\pi^*}(\mathbf{s}) \leq J^{\max} := \sum_{t=0}^{T} \gamma^t \cdot 0 = 0, \tag{6.4}$$

that is, any arbitrary policy will achieve a value less than or equal to the optimal policy and the optimal value can never be higher than the sum of maximum achievable rewards. The graphical model implies the marginal likelihood

$$\begin{aligned}
\mathrm{p}(\mathbf{J}^{\pi^*} \mid \mathbf{s}_0) &= \int \underbrace{\mathrm{p}(\mathbf{J}^{\pi^*} \mid \mathbf{J})}_{\text{Likelihood}} \underbrace{\mathrm{p}(\mathbf{J} \mid \boldsymbol{\pi}^*, \mathbf{s}_0, \mathbf{f})}_{\text{Trajectory}} \underbrace{\mathrm{p}(\boldsymbol{\pi}^*, \mathbf{f})}_{\text{System}} \, \mathrm{d}\mathbf{J} \, \mathrm{d}\boldsymbol{\pi}^* \, \mathrm{d}\mathbf{f} \, \mathrm{d}\mathbf{s}_0, && (6.5) \\
&\geq \int \mathrm{p}(\mathbf{J}^{\max} \mid \mathbf{J}) \, \mathrm{p}(\mathbf{J} \mid \boldsymbol{\pi}^*, \mathbf{s}_0, \mathbf{f}) \, \mathrm{p}(\boldsymbol{\pi}^*, \mathbf{f}) \, \mathrm{d}\mathbf{J} \, \mathrm{d}\boldsymbol{\pi}^* \, \mathrm{d}\mathbf{f} \, \mathrm{d}\mathbf{s}_0, && (6.6)
\end{aligned}$$

where the inequality is implied by (6.4) if the likelihood is monotonically decreasing with distance. Both the trajectory and the system terms have a clear interpretation, as they represent the recurrent structure and the function approximators, respectively. It is not clear, however, how to interpret the likelihood term. Given the value estimate $\mathbf{J}$, this term encapsulates an estimation of closeness to the optimal value and must encode that the optimal value is the highest possible value.

We now assume variational distributions $\mathrm{q}(\mathbf{f})$ and $\mathrm{q}(\boldsymbol{\pi}^*)$ introduced by the sparse GP formulations in Section 2.7. Analogously to the derivation of the DSVI-bound in (2.89) we get

$$\begin{aligned}
\log \mathrm{p}(\mathbf{J}^{\pi^*} \mid \mathbf{s}_0) &\geq \log \mathrm{p}(\mathbf{J}^{\max} \mid \mathbf{s}_0) \\
&= \log \int \mathrm{p}(\mathbf{J}^{\max} \mid \mathbf{J}) \, \mathrm{p}(\mathbf{J} \mid \boldsymbol{\pi}^*, \mathbf{s}_0, \mathbf{f}) \, \mathrm{p}(\boldsymbol{\pi}^*, \mathbf{f}) \, \mathrm{d}\mathbf{J} \, \mathrm{d}\boldsymbol{\pi}^* \, \mathrm{d}\mathbf{f} \, \mathrm{d}\mathbf{s}_0 \\
&\geq \mathbb{E}_{\mathrm{q}(\mathbf{s}_0, \dots, \mathbf{s}_T)} \left[\log \int \mathrm{p}(\mathbf{J}^{\max} \mid \mathbf{J}) \underbrace{\mathrm{p}(\mathbf{J} \mid \mathbf{s}_0, \dots, \mathbf{s}_T)}_{\text{Deterministic}} \, \mathrm{d}\mathbf{J} \right] - T \cdot \text{klterm} \\
&= \mathbb{E}_{\mathrm{q}(\mathbf{J})} [\log \mathrm{p}(\mathbf{J}^{\max} \mid \mathbf{J})] - T \cdot \text{klterm},
\end{aligned} \tag{6.7}$$

with $\text{klterm} = \mathrm{KL}(\mathrm{q}(\boldsymbol{\pi}^*) \parallel \mathrm{p}(\boldsymbol{\pi}^*)) + \mathrm{KL}(\mathrm{q}(\mathbf{f}) \parallel \mathrm{p}(\mathbf{f}))$ multiplied by T due to the recurrent structure. The distribution $\mathrm{q}(\mathbf{J})$ can easily be sampled from $\mathrm{q}(\mathbf{s}_0, \dots, \mathbf{s}_T)$ using the definition of the value function. A sample from $\mathrm{q}(\mathbf{s}_0, \dots, \mathbf{s}_T)$ can be drawn via the usual

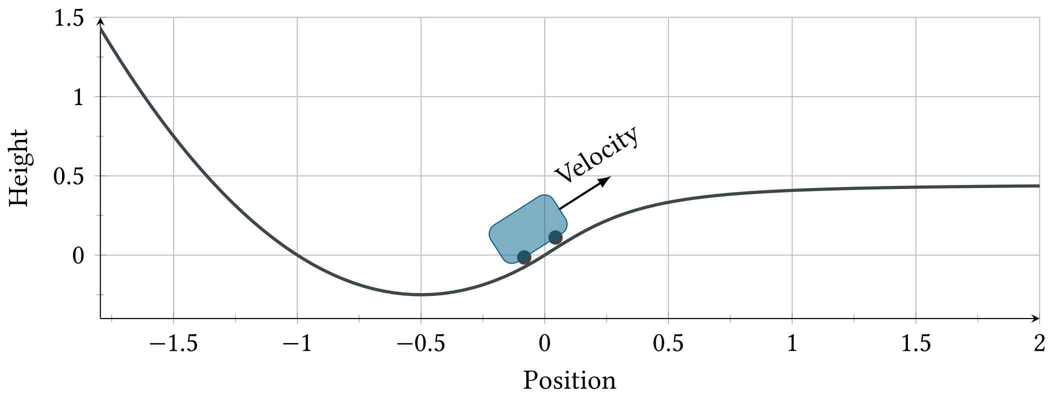


Figure 6.5: The core components of the mountaincar benchmark. The task is to drive an underpowered car up a mountain. The car has a one-dimensional position and velocity value and the agent can accelerate to the left or right. A successful policy must build momentum by first moving away from the goal to the right.

ancestral sampling scheme employed by DSVI. Assuming that the sufficient statistics assumption holds for both $\boldsymbol{\pi}^*$ and $\mathbf{f}$, the optimal policy still maximizes variational lower bound if $\mathrm{p}(\mathbf{J}^{\max} \mid \mathbf{J})$ is monotonous. We choose an exponential likelihood, that is

$$\begin{aligned} \mathrm{p}(\mathbf{J}^{\max} \mid \mathbf{J}) &\coloneqq \lambda \exp(-\lambda(\mathbf{J}^{\max} - \mathbf{J})) \\ &= \lambda \exp(\lambda \mathbf{J}), \end{aligned} \tag{6.8}$$

since $\mathbf{J}^{\max} = 0$. With $\lambda = 1$, the bound in (6.7) reduces to

$$\begin{aligned} &\mathbb{E}_{\mathrm{q}(\mathbf{J})}[\log \mathrm{p}(\mathbf{J}^{\max} \mid \mathbf{J})] - T \cdot \text{klterm} \\ &\quad = \mathbb{E}_{\mathrm{q}(\mathbf{J})}[\log \exp(\mathbf{J})] - T \cdot \text{klterm} \\ &\quad = \mathbb{E}_{\mathrm{q}(\mathbf{J})}[\mathbf{J}] - T \cdot \mathrm{KL}(\mathrm{q}(\boldsymbol{\pi}^*) \,\|\, \mathrm{p}(\boldsymbol{\pi}^*)) + \text{const.} \end{aligned} \tag{6.9}$$

This bound is similar to a standard policy iteration scheme but is also subject to the prior on $\boldsymbol{\pi}^*$.

As an example, we consider the mountain car system [79, 111]. Figure 6.5 visualizes the task of driving an underpowered car up a mountain. The car's engine is not strong enough to overcome the slope, and a successful agent must first drive away from the goal on the right to build momentum. The current state of the benchmark is described

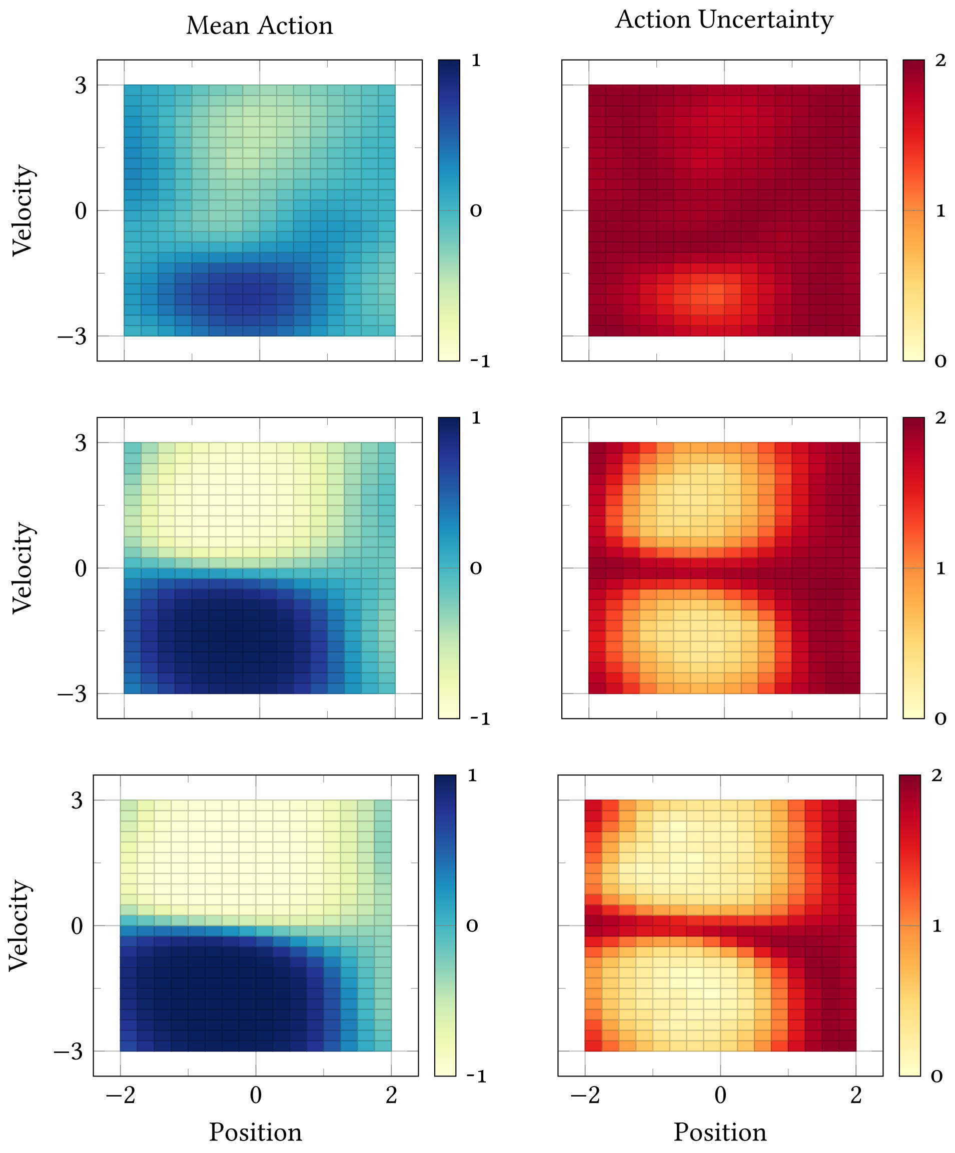


Figure 6.6: Bayesian belief about an optimal mountaincar policy after 1, 10 and 25 iterations (top to bottom). Starting off with an uninformed policy, the critical areas of the benchmark are quickly identified and the policy becomes certain about the correct action. Other areas remain uncertain, implying that the states are either never reached, or all actions lead to success.

by the one-dimensional position and velocity of the car, and the agent can accelerate the car in either direction.

We train a dynamics model with random observations of the system and maximize the variational bound in (6.7) to find an optimal policy to solve the benchmark problem. Figure 6.6 shows how the Bayesian belief about the optimal policy evolves during optimization of the bound. Starting off with an uninformed policy for which all actions are plausible in all states, the approximation of the optimal value is very uncertain and uninformed. During training, the areas in the input space that are relevant for solving the task emerge, and a successful policy is found that oscillates in the valley to build momentum and overcome the mountain. The inference scheme also uncovers areas in the input space, where uncertainty about the optimal policy is never reduced. Since this policy representation is used to parameterize the optimal value, these states are not relevant for a good approximation. In other words, these states are either never reached, or all actions lead to success.

Formulating policy search as an inference problem over the optimal value allows us to automatically uncover which parts of a system are relevant to solving a task. Similar to a surrogate in BO not needing to fully represent complicated structure away from the optimum, both a dynamics model and a policy representation need not understand a system fully to solve a reinforcement learning task. Separating a system in relevant and irrelevant parts to steer exploration or policy search are promising directions of research.

6.4 Future Work

Taking the step to apply machine learning in the physical world is one of the most important challenges for machine learning today. ML-systems that operate in safety-critical areas, interact with people, or carry responsibility must be robust, trustworthy, and assessable. This thesis explored how structured Gaussian process models can be used to formalize abstract knowledge about hierarchical systems and gain new insights from data. Future research will allow us to generalize these results and explore the following questions:

- How do we formulate data-efficient models together with domain experts?
- How do we ensure models can be trusted to take responsibility?
- How do we implement models to yield robust results?

Machine learning models can be effective tools for communication with experts if they allow an extensive inspection to achieve interpretability. Formulating principled generative models based on expert understanding allows us to reproduce abstract expectations, and combining such models with data allows us to gain new insights about the badly understood components of a system. In Sections 6.2 and 6.3 we discussed approaches to including downstream tasks in modeling problems. Bayesian learning in pipelines of tasks is an active area of study in probabilistic numerics [26], where multiple numerical algorithms are applied in succession. It is an interesting research question how these ideas can be implemented in more general problem settings where priors and expectations are less well defined.

In Bayesian inference, a model is typically evaluated with respect to a subjective prior belief and the likelihood of observations which are typically assumed to be independent. Hierarchical models are often formulated with components that have clear responsibilities to allow experts to independently reason about the prior assumptions of the different parts of a model. However, we have seen in Section 6.1 that once we include compositions of general function approximators in a generative model, strong dependency structures emerge, and reasoning about priors is difficult [43]. Future work will explore how to constrain hierarchical models beyond componentwise structural assumptions. Hierarchical models such as the alignment model in Chapter 4 can show rich and expert-interpretable structure in the shape of samples drawn from generative models. One interesting direction of research is to explore how to formalize expectations about such samples as a new way to place constraints on the internal structure of hierarchical models.

Interpretable models and principled uncertainty propagation often require costly computations. To implement models in practice, we need to rely on more efficient approximations, which can limit the range of representable solutions. As we have seen in Section 6.1, the current approaches to inference in deep Gaussian process models cannot represent complex hierarchical posteriors well, and more expressive alternatives tend to be computationally expensive. The search for efficient and expressive inference schemes for Gaussian processes is an active area of research [93, 98, 123] and a natural future direction of research is to ask how such schemes can be extended to hierarchical models. Another interesting question is how to combine the advantages of principled generative models with computationally efficient alternatives. Large parametric models such as Bayesian neural networks (BNN) allow large-scale inference and fast predictions at the cost of badly understood prior assumptions. Recent work [109] explored how to place Gaussian process priors on BNN models. It would be interesting to extend these ideas to more informative structural priors.

Bibliography

[1] Martín Abadi et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems". In: (2015).

[2] Mauricio Alvarez and Neil D. Lawrence. "Sparse convolved Gaussian processes for multi-output regression". In: *Advances in neural information processing systems*. 2009, pp. 57–64.

[3] Mauricio A. Alvarez, David Luengo, Michalis K. Titsias, and Neil D. Lawrence. "Efficient Multioutput Gaussian Processes through Variational Inducing Kernels." In: *AISTATS*. Vol. 9. 2010, pp. 25–32.

[4] Mauricio A. Alvarez, Lorenzo Rosasco, and Neil D. Lawrence. *Kernels for Vector-Valued Functions: a Review*. June 30, 2011. arXiv: `1106.6251 [cs, math, stat]`. URL: `http://arxiv.org/abs/1106.6251` (visited on 02/06/2017).

[5] Andrew Gelman. *Bayesian Data Analysis (Chapman & Hall/CRC Texts in Statistical Science)*. Chapman and Hall/CRC, Nov. 27, 2013.

[6] Karl J. Åström. *Introduction to Stochastic Control Theory*. Elsevier, Feb. 27, 1971. 318 pp. ISBN: 978-0-08-095579-7.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural machine translation by jointly learning to align and translate*. 2014. arXiv: `1409.0473`.

[8] Yaakov Bar-Shalom, Thomas E. Fortmann, and Peter G. Cable. "Tracking and Data Association". In: *The Journal of the Acoustical Society of America* 87.2 (Feb. 1, 1990), pp. 918–919. ISSN: 0001-4966. DOI: `10.1121/1.398863`.

[9] A. G. Barto, R. S. Sutton, and C. W. Anderson. "Neuronlike adaptive elements that can solve difficult learning control problems". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5 (Sept. 1983), pp. 834–846. ISSN: 0018-9472. DOI: `10.1109/TSMC.1983.6313077`.

[10] J. Bernardo, J. Berger, A. Dawid, and A. Smith. "Regression and classification using Gaussian process priors". In: *Bayesian statistics* 6 (1998), p. 475.

[11] Christopher Berner et al. *Dota 2 with large scale deep reinforcement learning.* 2019. arXiv: 1912.06680.

[12] Christopher M. Bishop. *Mixture density networks.* 1994.

[13] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Apr. 28, 2007.

[14] Eilyan Bitar and Pete Seiler. "Coordinated control of a wind turbine array for power maximization". In: *2013 American Control Conference.* 2013 American Control Conference. June 2013, pp. 2898–2904. DOI: 10.1109/ACC.2013.6580274.

[15] Erik Bodin, Neill D. F. Campbell, and Carl Henrik Ek. *Latent Gaussian Process Regression.* July 18, 2017. arXiv: 1707.05534 [cs, stat]. URL: http://arxiv.org/abs/1707.05534 (visited on 08/29/2017).

[16] Erik Bodin et al. "Modulating Surrogates for Bayesian Optimization". In: *Proceedings of the International Conference on Machine Learning (ICML) 119.* Feb. 24, 2020. arXiv: 1906.11152.

[17] Phillip Boyle, Marcus Frean, Phillip Boyle, and Marcus Frean. *Multiple output gaussian process regression.* 2005.

[18] Phillip Boyle and Marcus R. Frean. "Dependent Gaussian Processes." In: *NIPS.* Vol. 17. 2004, pp. 217–224.

[19] Leo Breiman. "Statistical modeling: The two cultures (with comments and a rejoinder by the author)". In: *Statistical science* 16.3 (2001), pp. 199–231.

[20] Greg Brockman et al. *OpenAI Gym.* June 5, 2016. arXiv: 1606.01540 [cs]. URL: http://arxiv.org/abs/1606.01540 (visited on 09/24/2018).

[21] Ching-An Cheng and Byron Boots. "Variational Inference for Gaussian Process Models with Linear Complexity". In: *Advances in Neural Information Processing Systems 30.* Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 5184–5194.

[22] Youngmin Cho and Lawrence K. Saul. "Kernel Methods for Deep Learning". In: *Advances in Neural Information Processing Systems 22.* Ed. by Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta. Curran Associates, Inc., 2009, pp. 342–350.

[23] Sungjoon Choi, Sanghoon Hong, and Sungbin Lim. *ChoiceNet: Robust Learning by Revealing Output Correlations.* May 16, 2018. arXiv: 1805.06431 [cs, stat]. URL: http://arxiv.org/abs/1805.06431 (visited on 08/23/2018).

[24] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. *Attention-Based Models for Speech Recognition*. June 24, 2015. arXiv: 1506.07503 [cs, stat]. URL: http://arxiv.org/abs/1506.07503 (visited on 08/17/2020).

[25] Timothy C. Coburn. *Geostatistics for natural resources evaluation*. Taylor & Francis Group, 2000.

[26] Jon Cockayne, Chris Oates, Tim Sullivan, and Mark Girolami. "Bayesian Probabilistic Numerical Methods". In: *SIAM Review* 61.3 (Jan. 2019), pp. 756–789. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/17M1139357. arXiv: 1702.03673.

[27] The Event Horizon Telescope Collaboration et al. "First M87 Event Horizon Telescope Results. III. Data Processing and Calibration". In: *The Astrophysical Journal Letters* 875.1 (Apr. 10, 2019), p. L3. ISSN: 2041-8205. DOI: 10.3847/2041-8213/ab0c57.

[28] The LIGO Scientific Collaboration and the Virgo Collaboration. "Observation of Gravitational Waves from a Binary Black Hole Merger". In: *Physical Review Letters* 116.6 (Feb. 11, 2016), p. 061102. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.116.061102. arXiv: 1602.03837.

[29] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. 2009.

[30] Ingemar J. Cox. "A review of statistical data association techniques for motion correspondence". In: *International Journal of Computer Vision* 10.1 (Feb. 1, 1993), pp. 53–66. ISSN: 1573-1405. DOI: 10.1007/BF01440847.

[31] Andreas Damianou. "Deep Gaussian processes and variational propagation of uncertainty". University of Sheffield, 2015.

[32] Andreas Damianou and Neil Lawrence. "Deep Gaussian Processes". In: *Artificial Intelligence and Statistics*. Artificial Intelligence and Statistics. Apr. 29, 2013, pp. 207–215.

[33] Andreas C. Damianou, Michalis K. Titsias, and Neil D. Lawrence. *Variational Inference for Uncertainty on the Inputs of Gaussian Process Models*. Sept. 8, 2014. arXiv: 1409.2287 [cs, stat]. URL: http://arxiv.org/abs/1409.2287 (visited on 09/05/2016).

[34] Amit Daniely, Roy Frostig, and Yoram Singer. *Toward Deeper Understanding of Neural Networks: The Power of Initialization and a Dual View on Expressivity*. Feb. 18, 2016. arXiv: 1602.05897 [cs, stat]. URL: http://arxiv.org/abs/1602.05897 (visited on 03/21/2018).

[35] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, Feb. 1, 2012. ISBN: 978-0-521-51814-7.

[36] Marc Deisenroth and Carl E. Rasmussen. "PILCO: A model-based and data-efficient approach to policy search". In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*. 2011, pp. 465–472.

[37] Marc Peter Deisenroth. "Efficient Reinforcement Learning using Gaussian Processes". KIT Scientific Publishing, 2010.

[38] Stefan Depeweg. "Modeling Epistemic and Aleatoric Uncertainty with Bayesian Neural Networks and Latent Variables". Dissertation. Munich: Technical University of Munich, 2019.

[40] Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. "Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning". In: *International Conference on Machine Learning*. 2018, pp. 1192–1201.

[41] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. *Learning and Policy Search in Stochastic Dynamical Systems with Bayesian Neural Networks*. May 23, 2016. arXiv: `1605.07127 [cs, stat]`. URL: `http://arxiv.org/abs/1605.07127` (visited on 02/19/2019).

[42] Simon Duane, Anthony D. Kennedy, Brian J. Pendleton, and Duncan Roweth. "Hybrid monte carlo". In: *Physics letters B* 195.2 (1987), pp. 216–222.

[43] David Duvenaud, Oren Rippel, Ryan P. Adams, and Zoubin Ghahramani. *Avoiding pathologies in very deep networks*. Feb. 24, 2014. arXiv: `1402.5836 [cs, stat]`. URL: `http://arxiv.org/abs/1402.5836` (visited on 09/21/2016).

[44] Bradley Efron. "Modern science and the Bayesian-frequentist controversy". In: (2005).

[46] G. Fubini. "Sugli integrali multipli". In: *Accademia dei Lincei, Rendiconti, V. Serie* 16.1 (1907), pp. 608–614. ISSN: 0001-4435.

[47] Yarin Gal. "Uncertainty in deep learning". In: *University of Cambridge* 1.3 (2016).

[48] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R (Springer Texts in Statistics)*. Springer, Aug. 11, 2013. ISBN: 978-1-4614-7137-0.

[50] Giulio D. Agostini. *Bayesian Reasoning in Data Analysis: A Critical Introduction*. World Scientific Pub Co Inc, Aug. 1, 2003. ISBN: 978-981-238-356-3.

[51] Javier González, Zhenwen Dai, Andreas Damianou, and Neil D. Lawrence. "Preferential Bayesian optimization". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70.* ICML'17. Sydney, NSW, Australia: JMLR.org, Aug. 6, 2017, pp. 1282–1291.

[52] Alexander Hans and Steffen Udluft. "Efficient uncertainty propagation for reinforcement learning with limited data". In: *International Conference on Artificial Neural Networks.* Springer, 2009, pp. 70–79.

[53] Marton Havasi, José Miguel Hernández-Lobato, and Juan José Murillo-Fuentes. *Inference in Deep Gaussian Processes using Stochastic Gradient Hamiltonian Monte Carlo.* June 14, 2018. arXiv: `1806.05490 [cs, stat]`. URL: `http://arxiv.org/abs/1806.05490` (visited on 06/21/2018).

[54] Daniel Hein. "Interpretable Reinforcement Learning Policies by Evolutionary Computation". Dissertation. Munich: Technical University of Munich, 2019. 160 pp.

[55] Daniel Hein et al. "A benchmark environment motivated by industrial control problems". In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI).* 2017 IEEE Symposium Series on Computational Intelligence (SSCI). Honolulu, HI: IEEE, Nov. 2017, pp. 1–8. ISBN: 978-1-5386-2726-6. DOI: `10.1109/SSCI.2017.8280935`.

[56] James Hensman, Nicolo Fusi, and Neil D. Lawrence. "Gaussian Processes for Big Data". In: *Uncertainty in Artificial Intelligence.* Citeseer, 2013, p. 282.

[57] James Hensman and Neil D. Lawrence. *Nested Variational Compression in Deep Gaussian Processes.* Dec. 3, 2014. arXiv: `1412.1370 [stat]`. URL: `http://arxiv.org/abs/1412.1370` (visited on 07/19/2017).

[58] James Hensman, Alexander G Matthews, Maurizio Filippone, and Zoubin Ghahramani. "MCMC for Variationally Sparse Gaussian Processes". In: *Advances in Neural Information Processing Systems 28.* Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 1648–1656.

[59] James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. "Scalable variational Gaussian process classification". In: *Journal of Machine Learning Research* 38 (2015), pp. 351–360.

[60] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[61] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. "Adaptive mixtures of local experts". In: *Neural computation* 3.1 (1991), pp. 79–87.

[62] Rodolphe Jenatton, Cedric Archambeau, Javier Gonzalez, and Matthias Seeger. "Bayesian optimization with tree-structured dependencies". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, Aug. 6, 2017, pp. 1655–1664.

[63] Melvin Johnson et al. "Google's multilingual neural machine translation system: Enabling zero-shot translation". In: *Transactions of the Association for Computational Linguistics* 5 (2017), pp. 339–351.

[64] Andre G. Journel and Ch J. Huijbregts. *Mining geostatistics*. Academic press, 1978.

[65] Markus Kaiser, Clemens Otte, Thomas A. Runkler, and Carl Henrik Ek. "Bayesian Alignments of Warped Multi-Output Gaussian Processes". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 6995–7004. arXiv: 1710.02766.

[66] Markus Kaiser, Clemens Otte, Thomas A. Runkler, and Carl Henrik Ek. "Bayesian Decomposition of Multi-Modal Dynamical Systems for Reinforcement Learning". In: *Neurocomputing* (Apr. 10, 2020). ISSN: 0925-2312. DOI: 10.1016/j.neucom.2019.12.132.

[67] Markus Kaiser, Clemens Otte, Thomas A. Runkler, and Carl Henrik Ek. "Data Association with Gaussian Processes". In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD) 2019*. Sept. 2019. arXiv: 1810.07158.

[68] Markus Kaiser, Clemens Otte, Thomas A. Runkler, and Carl Henrik Ek. "Interpretable Dynamics Models for Data-Efficient Reinforcement Learning". In: *Computational Intelligence and Machine Learning* ESANN 2019 proceedings (2019), p. 6.

[70] Diederik P Kingma, Tim Salimans, and Max Welling. "Variational Dropout and the Local Reparameterization Trick". In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 2575–2583.

[71] John R. Koza and John R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. Vol. 1. MIT press, 1992.

[72] Sascha Lange, Thomas Gabel, and Martin Riedmiller. "Batch reinforcement learning". In: *Reinforcement learning*. Springer, 2012, pp. 45–73.

[73] Neil D. Lawrence and Andrew J. Moore. “Hierarchical Gaussian process latent variable models”. In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 481–488.

[74] Miguel Lázaro-Gredilla. “Bayesian warped Gaussian processes”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 1619–1627.

[75] Miguel Lázaro-Gredilla, Steven Van Vaerenbergh, and Neil D. Lawrence. “Overlapping mixtures of Gaussian processes for the data association problem”. In: *Pattern Recognition* 45.4 (2012), pp. 1386–1395.

[76] Yann LeCun et al. “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4 (1989), pp. 541–551.

[77] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. *The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables*. Nov. 2, 2016. arXiv: 1611.00712 [cs, stat]. URL: http://arxiv.org/abs/1611.00712 (visited on 09/12/2018).

[78] Thomas P. Minka. “Expectation Propagation for approximate Bayesian inference”. In: *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2001, pp. 362–369. arXiv: 1301.2294.

[79] Andrew William Moore. “Efficient memory-based learning for robot control”. In: (1990).

[80] N. Morgan and H. Bourlard. “Generalization and Parameter Estimation in Feedforward Nets: Some Experiments”. In: *Advances in Neural Information Processing Systems 2*. Ed. by D. S. Touretzky. Morgan-Kaufmann, 1990, pp. 630–637.

[81] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, Aug. 24, 2012. 1098 pp. ISBN: 978-0-262-01802-9.

[82] Vilém Novák, Irina Perfilieva, and Jiri Mockor. *Mathematical principles of fuzzy logic*. Vol. 517. Springer Science & Business Media, 2012.

[83] C. J. Oates and T. J. Sullivan. *A Modern Retrospective on Probabilistic Numerics*. Jan. 14, 2019. arXiv: 1901.04457 [math, stat]. URL: http://arxiv.org/abs/1901.04457 (visited on 10/02/2019).

[84] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. “The matrix cookbook”. In: *Technical University of Denmark* 7 (2008), p. 15.

[85] Tobias Pfingsten, Malte Kuss, and Carl Edward Rasmussen. “Nonstationary gaussian process regression using a latent extension of the input space”. In: *Eighth World Meeting of the International Society for Bayesian Analysis (ISBA 2006)*. 2006.

[86] Carl E. Rasmussen and Zoubin Ghahramani. “Infinite Mixtures of Gaussian Process Experts”. In: *Advances in Neural Information Processing Systems 14*. Ed. by T. G. Dietterich, S. Becker, and Z. Ghahramani. MIT Press, 2002, pp. 881–888.

[87] Carl Edward Rasmussen. *Gaussian processes for machine learning*. 2006.

[88] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic Backpropagation and Approximate Inference in Deep Generative Models”. In: (Jan. 16, 2014).

[89] Martin Riedmiller. “Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method”. In: *European Conference on Machine Learning*. Springer, 2005, pp. 317–328.

[90] Simone Rossi, Markus Heinonen, Edwin V. Bonilla, Zheyang Shen, and Maurizio Filippone. *Rethinking Sparse Gaussian Processes: Bayesian Approaches to Inducing-Variable Approximations*. Mar. 9, 2020. arXiv: `2003.03080 [cs, stat]`. URL: `http://arxiv.org/abs/2003.03080` (visited on 03/14/2020).

[91] Cynthia Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5 (5 May 2019), pp. 206–215. ISSN: 2522-5839. DOI: `10.1038/s42256-019-0048-x`.

[92] Olga Russakovsky et al. *ImageNet Large Scale Visual Recognition Challenge*. Jan. 29, 2015. arXiv: `1409.0575 [cs]`. URL: `http://arxiv.org/abs/1409.0575` (visited on 08/17/2020).

[93] Hugh Salimbeni, Ching-An Cheng, Byron Boots, and Marc Deisenroth. *Orthogonally Decoupled Variational Gaussian Processes*. Sept. 24, 2018. arXiv: `1809.08820 [cs, stat]`. URL: `http://arxiv.org/abs/1809.08820` (visited on 10/16/2018).

[94] Hugh Salimbeni and Marc Deisenroth. “Doubly Stochastic Variational Inference for Deep Gaussian Processes”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4588–4599.

[95] J G Schepers and S P van der Pijl. “Improved modelling of wake aerodynamics and assessment of new farm control strategies”. In: *Journal of Physics: Conference Series* 75 (July 1, 2007), p. 012039. ISSN: 1742-6596. DOI: `10.1088/1742-6596/75/1/012039`.

[96] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. "Taking the Human Out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* 104.1 (Jan. 2016), pp. 148–175. ISSN: 0018-9219. DOI: 10.1109/JPROC.2015.2494218.

[97] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning.* 2014.

[98] Jiaxin Shi, Michalis K. Titsias, and Andriy Mnih. *Sparse Orthogonal Variational Inference for Gaussian Processes.* Feb. 29, 2020. arXiv: 1910.10596 [cs, stat]. URL: http://arxiv.org/abs/1910.10596 (visited on 03/03/2020).

[99] Galit Shmueli. "To explain or to predict?" In: *Statistical science* 25.3 (2010), pp. 289–310.

[100] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (Jan. 28, 2016), pp. 484–489. ISSN: 0028-0836. DOI: 10.1038/nature16961.

[101] Edward Snelson and Zoubin Ghahramani. "Sparse Gaussian processes using pseudo-inputs". In: *Advances in neural information processing systems.* 2005, pp. 1257–1264.

[102] Edward Snelson, Carl Edward Rasmussen, and Zoubin Ghahramani. "Warped gaussian processes". In: *Advances in neural information processing systems* 16 (2004), pp. 337–344.

[103] Edward Lloyd Snelson. "Flexible and efficient Gaussian process models for machine learning". Citeseer, 2007.

[104] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems 25.* Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 2951–2959.

[105] Jasper Snoek, Kevin Swersky, Richard S. Zemel, and Ryan P. Adams. *Input Warping for Bayesian Optimization of Non-stationary Functions.* Feb. 4, 2014. arXiv: 1402.0929 [cs, stat]. URL: http://arxiv.org/abs/1402.0929 (visited on 07/31/2017).

[106] Maryam Soleimanzadeh and Rafael Wisniewski. "Controller design for a wind farm, considering both power and load aspects". In: *Mechatronics* 21.4 (June 1, 2011), pp. 720–727. ISSN: 0957-4158. DOI: 10.1016/j.mechatronics.2011.02.008.

[107] The CMS Sollaboration and The LHCb Collaboration. "Observation of the rare Bso to Mu+ Mu- decay from the combined analysis of CMS and LHCb data". In: *Nature* 522.7554 (7554 June 2015), pp. 68–72. ISSN: 1476-4687. DOI: 10.1038/nature14474.

[108] Mervyn Stone. "Cross-validatory choice and assessment of statistical predictions". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 36.2 (1974), pp. 111–133.

[109] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. *Functional Variational Bayesian Neural Networks*. Mar. 13, 2019. arXiv: 1903.05779 [cs, stat]. URL: http://arxiv.org/abs/1903.05779 (visited on 12/13/2019).

[110] Richard S. Sutton. *The Bitter Lesson*. Mar. 13, 2019. URL: http://incompleteideas.net/IncIdeas/BitterLesson.html (visited on 08/14/2020).

[111] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. 2nd ed. Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 2018. 322 pp. ISBN: 978-0-262-19398-6.

[112] William M. Thorburn. "Occam's razor". In: *Mind* 24.2 (1915), pp. 287–288.

[113] Michalis K. Titsias. "Variational Learning of Inducing Variables in Sparse Gaussian Processes." In: *AISTATS*. Vol. 5. 2009, pp. 567–574.

[114] Michalis K. Titsias and Neil D. Lawrence. "Bayesian Gaussian process latent variable model". In: *International Conference on Artificial Intelligence and Statistics*. 2010, pp. 844–851.

[115] Volker Tresp. "Mixtures of Gaussian Processes". In: *Advances in Neural Information Processing Systems 13*. Ed. by T. K. Leen, T. G. Dietterich, and V. Tresp. MIT Press, 2001, pp. 654–660.

[116] Volker Tresp. "The wet game of chicken". In: *Siemens AG, CT IC 4, Technical Report* (1994).

[117] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, Nov. 10, 2013. ISBN: 978-0-387-21606-5.

[118] Ivan Ustyuzhaninov et al. "Compositional Uncertainty in Deep Gaussian Processes". In: *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*. Feb. 25, 2020. arXiv: 1909.07698.

[119] V. Vapnik. "Principles of Risk Minimization for Learning Theory". In: *Advances in Neural Information Processing Systems 4*. Ed. by J. E. Moody, S. J. Hanson, and R. P. Lippmann. Morgan-Kaufmann, 1992, pp. 831–838.

[120] Chunyi Wang and Radford M. Neal. *Gaussian Process Regression with Heteroscedastic or Non-Gaussian Residuals*. Dec. 26, 2012. arXiv: `1212.6246 [cs, stat]`. URL: `http://arxiv.org/abs/1212.6246` (visited on 04/16/2020).

[121] Edward Waring. "Vii. problems concerning interpolations". In: *Philosophical transactions of the royal society of London* 69 (1779), pp. 59–67.

[122] Andrew Gordon Wilson and Hannes Nickisch. *Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)*. Mar. 3, 2015. arXiv: `1503.01057 [cs, stat]`. URL: `http://arxiv.org/abs/1503.01057` (visited on 09/18/2017).

[123] James T. Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Peter Deisenroth. *Efficiently Sampling Functions from Gaussian Process Posteriors*. July 1, 2020. arXiv: `2002.09309 [cs, stat]`. URL: `http://arxiv.org/abs/2002.09309` (visited on 07/24/2020).

[124] Fariba Yousefi, Zhenwen Dai, Carl Henrik Ek, and Neil Lawrence. *Unsupervised Learning with Imbalanced Data via Structure Consolidation Latent Variable Model*. June 30, 2016. arXiv: `1607.00067 [cs, stat]`. URL: `http://arxiv.org/abs/1607.00067` (visited on 09/05/2016).

[125] Feng Zhou and Fernando De la Torre. "Generalized time warping for multimodal alignment of human motion". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1282–1289.

List of Figures

List of Tables