Universidad Autónoma de Madrid
Escuela Politécnica Superior
Departamento de Ingeniería Informática

Deep Gaussian Processes using Expectation Propagation and Monte Carlo Methods

Thesis presented to fulfill the requirements of the
Master's degree in Investigation and Innovation in Information and
Communications Technologies

By
Gonzalo Hernández Muñoz

under the supervision of
Daniel Hernández Lobato

Madrid, November 28, 2018

Contents

Contents ii

List of Figures v

List of Tables viii

1 Introduction 1
- 1.1 Motivation 1
- 1.2 Structure 2

2 Gaussian Processes 3
- 2.1 Gaussian Processes 3
- 2.2 Gaussian Process regression 4
  - 2.2.1 Predictive distribution for GP regression 6
  - 2.2.2 Marginal likelihood 6
  - 2.2.3 GPs with non-zero mean 7
- 2.3 Sparse Gaussian process approximation 7
  - 2.3.1 Selection of the inducing points locations 9
- 2.4 Summary 10

3 Approximate Inference 13
- 3.1 Variational Inference 13
- 3.2 Expectation Propagation 15
  - 3.2.1 Expectation Propagation algorithm 15
  - 3.2.2 Stochastic Expectation Propagation 17
- 3.3 Summary 18

4 Deep Gaussian Processes 21
- 4.1 The DGP model 22
  - 4.1.1 Deep sparse GPs 23
- 4.2 State of the art for inference in DGPs 24
  - 4.2.1 Fully Factorized DGPs Using Variational Inference 26
  - 4.2.2 DGPs for Regression Using Approximate Expectation Propagation 26
  - 4.2.3 Doubly Stochastic Variational Inference for DGPs 30
- 4.3 Deep Gaussian Processes Using Expectation Propagation and Monte Carlo Methods 33
  - 4.3.1 Predictive distribution 35
  - 4.3.2 Mean function 36

5 Experiments **39**
5.1 Regression problems . . . 40
5.2 Properties of the predictive distribution . . . 42
5.3 Big Data experiments . . . 44

6 Conclusions and Future Work **47**

Appendices **49**

A Gaussian identities **51**

Bibliography **51**

List of Figures

2.2.1 Comparison of samples from a GP prior (2.2.1) with different kernel functions. We can see that as the parameter ℓ of the RBF kernel the sampled function are less smooth 4
2.2.2 Samples taken from a GP prior comparing two of the most basic kernels. . . 5
2.2.3 Plot of the posterior distribution for GP regression. the red curve (dashed line) shows the original function from which the training points (in purple) were taken after adding some i.i.d Gaussian noise. The mean of the posterior distribution is shown in green (solid line) and the shaded blue area corresponds to the mean plus and minus two standard deviations. 7
2.3.1 GP regression using the FITC approximation, with starting inducing points in red and inducing points locations after maximizing marginal likelihood in green. Note that the y axis values of the inducing points don't represent anything as we are only interested in their locations (x axis). 10
2.4.1 Gaussian process regression on Mauna Loa CO2 data. This dataset is non-stationary as its mean is increasing with time. Using a GP in this problem requires a covariance function composed of several different kernels (RBF, periodic kernel, rational quadratic, etc.). One drawback of this approach is the increased computational complexity or choosing the right kernels without prior information from the problem. The code to generate this figure has been extracted from [Pedregosa et al., 2011] and [Rasmussen and Williams, 2006]. 11

3.1.1 Visualization of the variational inference lower bound. Because the log likelihood $p(\mathcal{D})$ has a constant value for any distribution q and the KL divergence is always positive, we can see that maximizing the lower bound is equivalent to minimizing the KL divergence. Figure extracted from [Bishop, 2006]. . . 15
3.3.1 Comparison of the different minimizations of the KL divergence. A Gaussian mixture (in blue) is approximated by a single Gaussian (in red). (a) shows the result of minimizing the forward KL divergence. (b) and (c) shows the results of minimizing the reverse KL divergence with different initializations. See the text for more details. Figures extracted from [Bishop, 2006]. 19

4.0.1 A standard deep neural network (NN) architecture is shown on the left. Compare that with the single hidden layer NN corresponding to a shallow GP (middle) and the deep NN with two hidden layers corresponding to a DGP (right). A GP is equivalent to a NN with infinite units in the hidden layers where the output of said layers are random variables (as expected by the GP definition). Images are based in the work from Duvenaud et al. [2014]. 21

4.0.2 Samples taken from a DGP model with two layers using RBF kernel for both layers. The two plots at the top correspond to samples from the GP priors on each layer respectively. The figure at the bottom represents samples taken from the composition of both GPs, that is; the output of the first layers is feed as the input to the second layer. One sample in each figure is highlighted to improve readability. As we can see in the bottom picture, the DGP samples are complex functions in which the lengthscales do not remain constant trough the whole x axes. 22

4.1.1 DGP example for a model with two layers $L = 2$. The nodes on each layer show the corresponding GP prior from Eq. (4.1.3). The 2-dimensional training points (on the left) are feed to the first layer (hidden) of the network, with $D_1 = 2$, the output at each layer is contaminated with some Gaussian noise (4.1.4). 23

4.1.2 Graphical model for the Deep sparse GP with $L = 2$. Observe that now the hidden layers not only depend on the output of the layer about but also on the inducing points variables. 24

4.2.1 Comparison of the full DGP model shown in Section 4.1.1 (left) with the model using a fully factorized variational posterior from Section 4.2.1. Note that this model ignores the important correlations between layers by making the input of a layer independent of the layer above. 27

4.2.2 Procedure for computing $\ln \mathcal{Z}$ for a single point in a DGP model with 2 layers. At the first layer, $q(h^1)$ is given by a Gaussian distribution (top graph in blue) so it can be calculated in closed form. At the second layer the true distribution for $\ln \mathcal{Z}$ (in blue) is no longer a Gaussian and its given by (4.2.15). The distribution can be approximated by a Gaussian with the same moments (in orange) as explained in (4.2.21). 29

4.3.1 8000 samples taken from $q(y|h^1)$ in Eq.(4.3.8) for different inputs x, the plots show histograms of the probability of the output y. 34

4.3.2 Procedure for computing $\ln \mathcal{Z}$ for a single point in a DGP model with 2 layers. At the first layer, $q(h^1)$ is given by a Gaussian distribution (top graph in blue) from which we sample from. At the second layer the true distribution for $\ln \mathcal{Z}$ (in blue) is no longer a Gaussian and its given by (4.3.10). Our proposed approach calculates and propagates samples through the network (in green) which is expected to make the model more flexible by being able to work with complex distributions. 35

4.3.3 Comparison of the two methods explained in this thesis to calculate $\ln \mathcal{Z}_n$ for a DGP model with 2 layers. The DGP-AEP method proposed in [Bui et al., 2016] approximates the output distribution $q(y|h^1)$ with a Gaussian distribution (in red). The histogram of the samples from our method to calculate $\ln \mathcal{Z}$ are shown in green. We see that, in this case, the Gaussian approximation fails to capture the bimodality from distribution $q(y|h^1)$. . . 36

4.3.4 Predictive distribution for a DGP with 3 layers for the step function problem. The plot shows that the samples (in purple) taken outside of the training $\mathbf{x}$ locations (in green) are generated from a multimodal distribution. 37

4.3.5 Samples taken from the DGP-AEPMCM model with 6 layers, it is possible to see that the functions generated by the DGP model are non-injective, making nearly impossible to recover the original x value from a given y value. Problems can arise during the training phase due to this pathology. One sample is highlighted to improve readability. 38

5.1.1 Test RMSE values plotted with standard errors for all the UCI datasets. Our proposal in brown with a diamond marker for the mean. Lower, to the left is better. Results for the DGP-AEP model using the protein dataset are missing due to excessive computation time. 40
5.1.2 Test Log-Likelihood values plotted with standard errors for all the UCI datasets. Our proposal in brown with a diamond marker for the mean. Higher, to the right is better. Results for the DGP-AEP model using the protein dataset are missing due to excessive computation time. 42
5.2.1 Comparison of samples from the predictive distribution for the DGP-VI model and DGP-AEPMCM model. The left figure shows the functions (in red) from which the training data (in blue) was generated. The middle figure shows samples of the predictive distribution for the DGP-VI model and the right figure shows samples from the predictive distribution for the DGP-AEPMCM model. Both posterior distributions will yield similar RMSE results, but the DGP-VI model is not able to capture the bimodal nature of the problem. 43
5.2.2 Comparison of samples from the predictive distribution for the DGP-VI model and DGP-AEPMCM model. The left figure show the function (in red) from which the training data (in blue) was generated. Observe that the noise does not remain constant, creating areas where the noise is significantly reduced. The middle figure shows samples of the predictive distribution for the DGP-VI model and the right figure shows samples from the predictive distribution for the DGP-AEPMCM model. 44
5.3.1 Results for RMSE (Left, the lower the better) and Log-likelihood (right, the higher the better) for the dataset Year, against training time (in seconds, log scale) measured every 100 gradient steps. 45
5.3.2 Results for RMSE (Left figure, the lower the better) and Log-likelihood (right figure, the higher the better) for the Airline dataset, against training time (in seconds, log scale) measured every 100 gradient steps. 46

List of Tables

5.0.1 Summary of the UCI datasets used in our experiments. 39
5.1.1 Regression RMSE (the lower the better) results. Mean for 20 splits and standard errors (in parentheses). 41
5.1.2 Regression test log-Likelihood (the higher the better) results. Mean for 20 splits and standard errors (in parentheses). 41
5.2.1 Results for the sin-cos synthetic dataset 43
5.2.2 Results for the heteroscedastic noise synthetic dataset 44
5.3.1 Results for the Big data experiments. Year 45
5.3.2 Results for the Big data experiments. Airline 46

Resumen

El aprendizaje automático puede definirse como una aplicación de la inteligencia artificial que proporciona a los sistemas la habilidad de aprender y mejorar a partir de la experiencia. Una de las principales áreas de trabajo de este campo es el aprendizaje supervisado, en el que se intenta predecir una variable de salida a partir de unas variables de entrada.

Este tipo de problemas se dividen en varias fases. En la primera se provee al sistema de los denominados datos de entrenamiento. Estos datos incluyen tanto las variables de entrada como las de salida (etiquetas). Tras la fase de entrenamiento el sistema está listo para realizar predicciones precisas sobre las variables de salida a partir de las variables de entrada sin etiquetar. Dependiendo del tipo de variables de salida del problema se puede diferenciar entre problemas de clasificación (variables categóricas) o problemas de regresión (variables continuas). Dentro del aprendizaje automático, los procesos Gaussianos son modelos no paramétricos que presentan numerosas ventajas con respecto a otros modelos.

Al ser métodos bayesianos, proporcionan no solamente una predicción del valor de la variable de salida sino también un grado de incertidumbre para la misma. Tampoco es necesario realizar suposiciones sobre la forma del proceso (función) que generó los datos. Es suficiente con incluir información de alto nivel (suavidad, periodicidad, etc.) en la llamada función de covarianzas o kernel que, junto con la media, definen el proceso Gaussiano. A pesar de ser métodos robustos, resistentes al sobre-aprendizaje, los procesos Gaussianos quedan limitados por la expresividad de la función de covarianzas. Por ello, se han intentado extender estos modelos tradicionales a variantes más expresivas, por ejemplo, considerando funciones de covarianza más sofisticadas o integrándolos en estructuras probabilísticas más complejas. Sin embargo, ninguno de estos enfoques lleva a arquitecturas profundas.

Recientemente se ha mostrado que los procesos Gaussianos pueden superar estos problemas y usarse como unidades individuales para construir redes profundas, dando como resultado procesos Gaussianos profundos. Estos modelos mantienen las ventajas de los procesos Gaussianos estándar, pero reducen las hipótesis realizadas sobre los datos, conduciendo a modelos más flexibles. La contrapartida de estos modelos es que presentan dificultades a la hora de entrenarlos, pues la inferencia bayesiana exacta no es posible y se han de usar técnicas de inferencia aproximada. Algunos modelos que representan el estado de arte en la investigación sobre procesos Gaussianos profundos han intentado hacer uso de las técnicas de inferencia aproximada, como por ejemplo, Inferencia Variacional, o variantes aproximadas del algoritmo de propagación de esperanzas.

En este trabajo de fin de máster presentamos un nueva técnica para realizar inferencia en procesos Gaussianos profundos mediante el uso de un método de inferencia aproximada basado en métodos Montecarlo y el algoritmo de propagación de esperanzas. Demostramos mediante exhaustivos experimentos que nuestro enfoque proporciona resultados competitivos al nivel de otros modelos que representan el estado del arte.

Por último mostramos que el modelo propuesto escala bien para conjuntos de datos grandes y su uso es adecuado para la resolución de problemas de Big Data. Además, presenta otras propiedades que analizamos en nuestros experimentos, como la posibilidad de capturar ruido dependiente de los datos de entrada o el modelado de distributiones predictivas multimodales. Estas propiedades no son observadas en otros métodos de inferencia aproximada.

Abstract

Machine learning can be defined as the application of artificial intelligence that provides systems with the ability of learning and improve from experience. One of the main research areas in this field is supervised learning, in which an output variable is predicted from some input variables.

These types of problems are divided into several stages. In the first one, the system is provided with the so-called training data. This data includes both input and (labeled) output variables. After the training phase, the system is ready to make accurate predictions about the values of the output variables that are not labeled. Depending on the type of output variables, we can differentiate between classification (categorical variables) and regression (continuous variables) problems.

Gaussian processes are non-parametric machine learning models that present advantages over other models. They provide not only a prediction about the output value, but they also provide the uncertainty of such prediction. It is also not required to make assumptions about the form of the process (function) that generated the data, is enough to include the high-level information (smoothness, periodicity, ...) in the so-called covariance function or kernel that jointly with the mean define the model. Although Gaussian processes are robust to over-fitting they are limited by the expressiveness of the covariance function. There has been some work trying to extend this traditional model to other more expressive variants, by for example considering more sophisticated covariance functions or integrating the model in more complex probability structures. However, none of these approaches make use of deep architectures.

Recently it has been shown that Gaussian processes can overcome these problems and be used as individual units to construct deep networks giving rise to deep Gaussian processes. These models maintain the advantages of single layer Gaussian processes but reduce the hypotheses made about the data, yielding more flexible models. Deep Gaussian processes have proven hard to train because exact Bayesian inference is not possible and approximate inference techniques have to be used. Some models that represent the state of the art on deep Gaussian processes research have used some of these techniques like variational inference or approximate variants of the expectation propagation algorithm.

In this master's thesis, we present a new machine learning model for inference in deep Gaussian processes models using an approximate inference technique based on Monte Carlo methods and the expectation propagation algorithm. We demonstrate through extensive experiments that our approach provides competitive results even when compared with some state of the art models.

Finally, we show that our model scales well with bigger datasets and it is suitable to use the proposed approach to solve Big Data problems. Furthermore, our model presents different properties such as being able to capture noise that is dependent on the input and modeling multimodal predictive distributions. Both of these properties, which are not shared with other approximate inference methods, are analyzed in our experiments.

Acknowledgements

Completion of this master's thesis has required not only my own efforts but efforts from many people around me. First and foremost, I would like to thank my supervisor Daniel. His help has been invaluable, investing many hours in meeting with me week after week. This thesis would not have been possible without him. I would also like to thank all the people at UAM that contributed to this work: the Machine Learning research group whose passion has been of great inspiration through the years and the "Centro de Computacion Cientifica (CCC)", that granted me the all the computing days I needed to finish this work. Last but not least, I would like to thank all the people around me, especially my family, for whom the area of knowledge of this thesis may be unknown but they still made the most valuable contribution: support and unconditional love. To all of them, thank you.

Chapter 1

Introduction

1.1 Motivation

Machine learning is being included in every aspect of our day to day life: facial recognition, medical diagnosis, recommendation engines, language processing, and the list keeps growing. A lot of the tasks mentioned can be solved using supervised learning methods in which part of the data given to the model are tagged so the model can find patterns in them during the "training phase" and be able to tag new unobserved data.

With so much competition in the field, new models are created every day to try to improve the state of the art algorithms. When comparing models, the first criteria that are usually used to evaluate the performance of a model are metrics like root mean squared error (RMSE) in regression and accuracy in classification tasks, but as the new models keep improving and datasets grow in size, other characteristics such as training time or the number of parameters that need to be tuned gain more importance.

The model proposed in this master's thesis is an improvement over other deep Gaussian process models. Results show that the quality of the predictive distribution for our proposal provides higher test log-likelihood values, which means that our model gives a better explanation of how the data was generated. However, our experiments not only focus on the metrics that evaluate a model performance but also on how to improve in different areas like efficiency.

When comparing deep Gaussian processes against other machine learning models, we see that they have numerous advantages like providing a confidence interval about a prediction instead of a single point estimation. Many real-world problems can be solved using a range of equally probable estimations rather than a single prediction.

Gaussian processes (GPs) are known to be models that do not scale well with an increasing number of data points. As one can expect, deep Gaussian processes have the same problem, which is worsened when working with deep models that have too many layers or a high number of dimensions in each layer. This makes efficient approximations crucial to using Deep GP models in the future. Our proposed method does not suffer from these limitations and we prove through our experiments that it can be used with larger datasets and with deep architectures with a high number of hidden dimensions.

Our work also improves the difficult phase of choosing the model parameters. Shallow Gaussian processes need to be provided with a kernel and its parameters. While kernel parameters can be learned from data, the specific kernel functions are hard to learn from data automatically and need to be chosen from a wide range of different ones. Deep Gaussian processes are more flexible and reduce the problems that could occur if the wrong kernel function is chosen. This also presents a great advantage over other methods that need to find the model parameters by grid search techniques which scale poorly with the number of model parameters as it increases the size of the grid.

1.2 Structure

This master's thesis is divided into six main chapters which can be grouped into three different parts.

The first part, composed of chapters 2 and 3, covers the theory and needed background around Gaussian Processes (GPs) and approximate inference techniques that are used in following chapters and in the main proposal of this thesis. We show the properties, advantages and how GPs can be used with large datasets. Finally, we explain the modeling limitations of GPs which are the motivation for the research around Deep Gaussian Processes (DGPs). Because doing inference in DGPs is intractable, approximate inference methods are needed. We explain two of the most used methods and the advantages and drawbacks of each of them in the third chapter.

The second part includes chapter 4 and it covers the main contribution of this thesis to the state of the art algorithms in machine learning and especially in DGPs. We show how a DGP is defined and what are the advantages over shallow GP models. At the same time we explain what makes training DGP models challenging and what different approaches have been taken in the literature. We explain our proposal and how it compares to the other state of the art techniques and what are the expected results when comparing with them.

The last part includes chapters 5 and 6. We explain the procedure followed in our three main experiments covering different interest areas like metrics comparison, handling big data problems and the modeling properties of our proposal. The last chapter includes the conclusions and the main lines for future work.

Chapter 2

Gaussian Processes

In this chapter we present the basic theory for Gaussian processes. A Gaussian process (GP) is often defined as a distribution over functions. Intuitively, we can think of a GP as the extension of the Gaussian distribution to the infinite dimensional case in which each sample will represent a function. It might seem difficult to work with a distribution over the infinite space of functions; however, we only need to consider the values of the function at the set of input values that we are interested in. Usually, these values are the ones corresponding to the training and test data points. When using GPs as machine learning models we will be working in a finite space.

2.1 Gaussian Processes

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution [Rasmussen and Williams, 2006]. It is defined by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$ and can be written as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2.1.1}$$

with

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{2.1.2}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathrm{Cov}(f(\mathbf{x}), f(\mathbf{x}')) = \mathbb{E}[(f(\mathbf{x}) - \mathbb{E}[f(\mathbf{x})])(f(\mathbf{x}') - \mathbb{E}[f(\mathbf{x}')])] \tag{2.1.3}$$

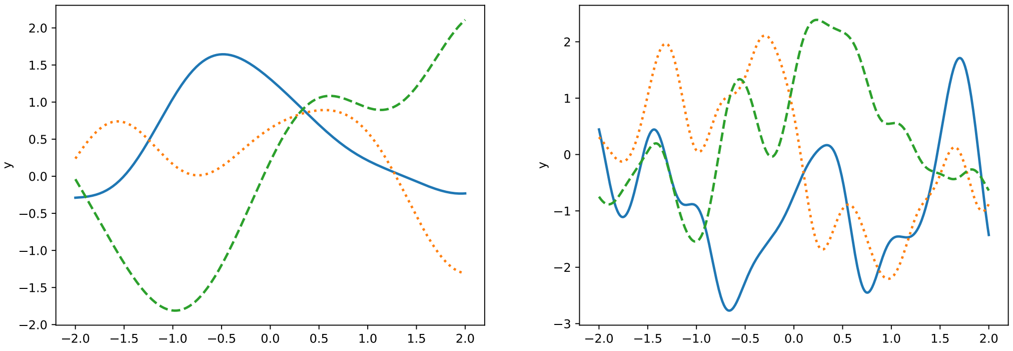
We usually take the mean to be zero but, as shown in Section 2.2.3, this need not be the case. $k(\mathbf{x}, \mathbf{x}')$ can be thought as a similarity function between the inputs of the problem. It has to be a valid kernel function so it has to satisfy the symmetry requirement $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$ and the matrix:

$$\begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \dots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

should be positive semi-definite. In general, if a kernel satisfies the Mercer's theorem it can be expressed as an inner product [Murphy, 2012]

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')\,, \tag{2.1.4}$$

mapping an arbitrary D-dimensional input $\mathbf{x}$ to a scalar value. As we will see in Section 2.2 the kernel function does not depend on the data but it can include some additional parameters to define the properties of the functions that can be drawn and explained by a GP.



(a) Samples of a GP prior with RBF kernel and $\ell = 1.1$.

(b) Samples of a GP prior with RBF kernel and $\ell = 0.3$.

Figure 2.2.1: Comparison of samples from a GP prior (2.2.1) with different kernel functions. We can see that as the parameter ℓ of the RBF kernel the sampled function are less smooth

2.2 Gaussian Process regression

The GP regression model is a Bayesian approach which assumes that the function we want to model takes values according to the prior

$$p(\mathbf{f}|\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}) = \mathcal{GP}(\mathbf{0}, k(\cdot,\cdot)) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})\,, \tag{2.2.1}$$

where $\mathbf{f}$ is the vector of function values and $\mathbf{K}$ is the kernel function evaluated at every pair of input points $\{\mathbf{x}_n\}_{n=1}^{N}$. The notation $\mathcal{N}(\mathbf{f}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ indicates that $\mathbf{f}$ is distributed as a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. An example of a widely used kernel is the squared exponential kernel, also called Radial Basis Function kernel (RBF):

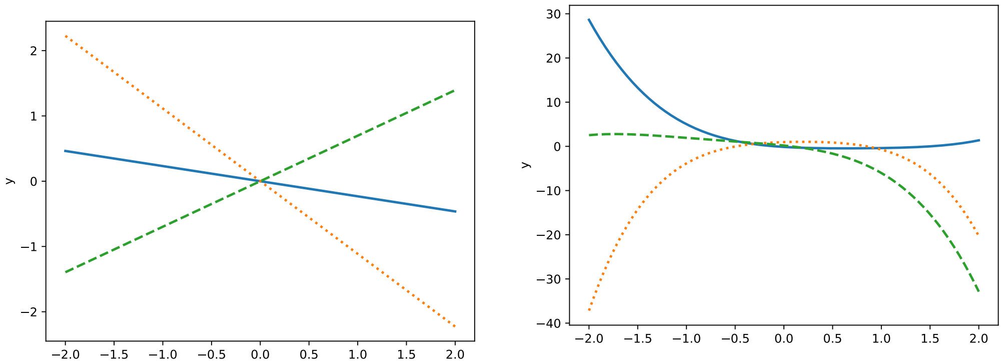
$$k_{\text{rbf}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left\{-\frac{(\mathbf{x} - \mathbf{x}')^2}{2\ell^2}\right\}\,. \tag{2.2.2}$$

As shown in Eq. (2.2.2) it only has two parameters that control the properties of the function [Duvenaud, 2014]:

- ℓ determines how abruptly our data changes values, higher values produce prior functions that smooth. Sometimes this parameters can be expressed independently for each dimension of the problem. This approach is commonly known as automatic relevance determination or ARD [Neal, 1996a] [MacKay, 1996]
- σ determines the scale of our data, a bigger scale factor increases the distance that the data moves away from the mean.

In Figures 2.2.1b and 2.2.1b we show samples taken from the GP prior in (2.2.1) with different parameters for the RBF kernel.

Some other examples of kernels include the polynomial kernel (2.2.4), with two parameters: c and d, that control the family of polynomial curves that can be sampled from the GP prior (2.2.1). The linear kernel (2.2.3): which is just a special case of the polynomial



(a) Samples of a GP prior with linear kernel.

(b) Samples of a GP prior with a polynomial kernel with $d = 4$.

Figure 2.2.2: Samples taken from a GP prior comparing two of the most basic kernels.

kernel is given by the inner product (plus a constant) of $\mathbf{x}$ and $\mathbf{x}'$. Samples from both kernels can be seen in Figures 2.2.2a and 2.2.2b. The expression for the linear kernel is:

$$k_{\text{linear}}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}' + c \,, \tag{2.2.3}$$

$$k_{\text{poly}}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + c)^d \,. \tag{2.2.4}$$

In a regression problem we are given a training set of N pairs of inputs $\mathbf{x}$ and their corresponding observed outputs values $\mathbf{y}$. In this setting, we suppose that the output values are generated by some function (or process) $f(\mathbf{x})$ that we want to model. Usually, the inputs $\{\mathbf{x}_i\}_{i=1}^N$ are D-dimensional vectorial values and the observations $\{y_i\}_{i=1}^N$ are contaminated with some random (Gaussian) noise ϵ. We can write that as:

$$y_i = f(\mathbf{x}_i) + \epsilon \,, \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}}) \,, \tag{2.2.5}$$

where $\mathbf{X}$ is a $N \times D$ matrix. We want to compute the output values $\mathbf{f}_\star$ for that process at the input test locations $\mathbf{x}_\star$.

By the definition of a GP, the joint distribution of the training values $\mathbf{f}$ and the noise-free test points $\mathbf{f}_\star$ are jointly a Gaussian and, under this model, the likelihood $p(\mathbf{y}|\mathbf{f})$ is also a Gaussian [Quiñonero-Candela and Rasmussen, 2005]:

$$p(\mathbf{f}, \mathbf{f}_\star) = \mathcal{N}\left(\mathbf{f}, \mathbf{f}_\star \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{\mathbf{f},\mathbf{f}} & \mathbf{K}_{\star,\mathbf{f}} \\ \mathbf{K}_{\mathbf{f},\star} & \mathbf{K}_{\star,\star} \end{bmatrix}\right) \,, \tag{2.2.6}$$

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2_{noise}\mathbf{I}) \,, \tag{2.2.7}$$

where $\mathbf{I}$ is the identity matrix and the covariance matrices are given by the kernel between the function values at test points $\mathbf{f}_\star$ and train points $\mathbf{f}$:

$$\begin{aligned} \mathbf{K}_{\mathbf{f},\mathbf{f}} &= k(\mathbf{X}, \mathbf{X}) \,, \quad & \mathbf{K}_{\star,\mathbf{f}} &= k(\mathbf{X}_\star, \mathbf{X}) \,, \\ \mathbf{K}_{\mathbf{f},\star} &= k(\mathbf{X}, \mathbf{X}_\star) \,, \quad & \mathbf{K}_{\star,\star} &= k(\mathbf{X}_\star, \mathbf{X}_\star) \,. \end{aligned} \tag{2.2.8}$$

Here $k(\mathbf{X}, \mathbf{X})$ represents the kernel function evaluated between every pair of training points $\mathbf{X}$. $k(\mathbf{X}, \mathbf{X}_\star)$ is the kernel evaluated between the pair of training $\mathbf{X}$ and test points $\mathbf{X}_\star$ and satisfies $\mathbf{K}_{\mathbf{f},\star} = {\mathbf{K}_{\star,\mathbf{f}}}^T$. Finally, $k(\mathbf{X}_\star, \mathbf{X}_\star)$ is the kernel between pairs of test points. It is possible to absorb the noise and write the joint distribution of the observed values and the latent function values as:

$$p(\mathbf{y}, \mathbf{f}_\star) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_\star\end{bmatrix}\middle|\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{K}_{\mathbf{f},\mathbf{f}} + \sigma^2_{\text{noise}} I & \mathbf{K}_{\star,\mathbf{f}}\\ \mathbf{K}_{\mathbf{f},\star} & \mathbf{K}_{\star,\star}\end{bmatrix}\right). \tag{2.2.9}$$

2.2.1 Predictive distribution for GP regression

As we are interested in the predictive distribution for the test points we can calculate the conditional distribution $p(\mathbf{f}_\star|\mathbf{y})$ in closed form by using the Gaussian identities (see Appendix A):

$$p(\mathbf{f}_\star|\mathbf{y}) = \mathcal{N}(\mathbf{f}_\star|\boldsymbol{\mu}_\star, \boldsymbol{\Sigma}_\star), \tag{2.2.10}$$

where

$$\boldsymbol{\mu}_\star = \mathbf{K}_{\star,\mathbf{f}}(\mathbf{K}_{\mathbf{f},\mathbf{f}} + \sigma^2_{noise}\mathbf{I})^{-1}\mathbf{y}, \tag{2.2.11}$$

$$\boldsymbol{\Sigma}_\star = \mathbf{K}_{\star,\star} - \mathbf{K}_{\star,\mathbf{f}}(\mathbf{K}_{\mathbf{f},\mathbf{f}} + \sigma^2_{noise}\mathbf{I})^{-1}\mathbf{K}_{\mathbf{f},\star}. \tag{2.2.12}$$

Note that $\mathbf{X}_\star$ can represent either a single point or multiple test points and the equation (2.2.10) will remain unchanged.

2.2.2 Marginal likelihood

Another interesting term which can be used for model comparison is the marginal likelihood $p(\mathbf{y})$. To calculate it we can simply marginalize $\mathbf{f}$ (the latent variables) from the product of the prior (2.2.1) and the likelihood $p(\mathbf{y}|\mathbf{f})$ in (2.2.7). Once again we can make use of the Gaussian identities in Appendix A to calculate the integral. Because all the terms are Gaussian, we have:

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f} = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{\mathbf{f},\mathbf{f}} + \sigma^2_{\text{noise}} I). \tag{2.2.13}$$

This quantity represents the probability of observing the data $\mathbf{y}$ in the $\mathbf{x}$ locations assuming that the latent variables $\mathbf{f}$ have been generated from the GP prior and contaminated with Gaussian noise with variance σ^2_{noise}. Usually, when making model selection we will be interested in the log marginal likelihood instead:

$$\log p(\mathbf{y}) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{K}_{\mathbf{f},\mathbf{f}} + \sigma^2_{noise} I| - \frac{1}{2}\mathbf{y}^T(\mathbf{K}_{\mathbf{f},\mathbf{f}} + \sigma^2_{noise} I)^{-1}\mathbf{y}. \tag{2.2.14}$$

This quantity can be used to tune the model parameters: the kernel parameters. Because the marginal likelihood is a smooth function of the hyperparameters we simply have to maximize it by calculating its gradients with respect to these parameters. This presents an advantage of Gaussian processes as opposed to other machine learning models in which model parameters are tuned via cross-validation; requiring to tune and train the model multiple times hence increasing the computational complexity.

Figure 2.2.3 shows an example of a GP regression model for a function $f(\mathbf{x})$ and some training points. We can see, as expected, that the variance of the predictive distribution is reduced to near 0 around the training examples, making all samples drawn from the GP posterior lie close to this area. In the noise free case the variance on the training examples will be zero.

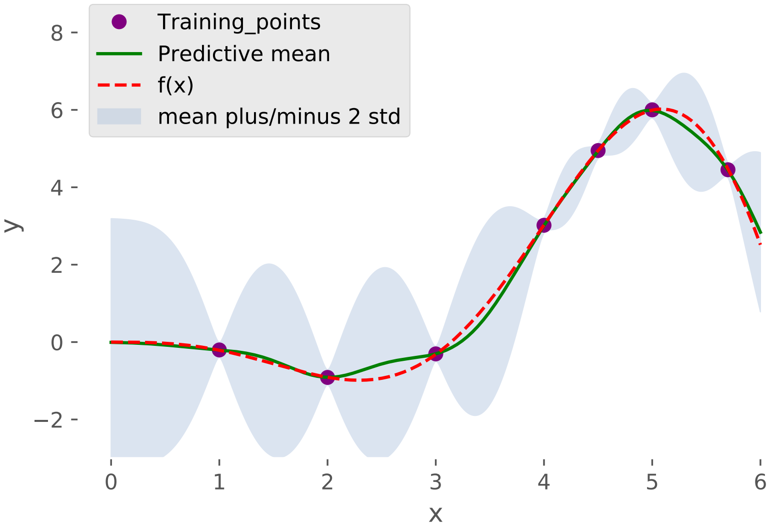


Figure 2.2.3: Plot of the posterior distribution for GP regression. the red curve (dashed line) shows the original function from which the training points (in purple) were taken after adding some i.i.d Gaussian noise. The mean of the posterior distribution is shown in green (solid line) and the shaded blue area corresponds to the mean plus and minus two standard deviations.

2.2.3 GPs with non-zero mean

In Section 2.2 we showed how to use the GP regression model with the mean term for the prior (2.2.1) set to zero. This choice does not constrain the mean of the posterior distribution to be zero. However, as we will show in further chapters of this thesis this may not be the right choice for Deep Gaussian processes. To add a fixed mean function $m(\mathbf{x})$ we modify the joint distribution in (2.2.9) as follows:

$$p(\mathbf{y}, \mathbf{f}_\star) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{f}_\star\end{bmatrix}\middle|\begin{bmatrix}m(\mathbf{x})\\ m(\mathbf{x}_\star)\end{bmatrix}, \begin{bmatrix}\mathbf{K}_{\mathbf{f},\mathbf{f}} + \sigma^2_{noise} I & \mathbf{K}_{\star,\mathbf{f}}\\ \mathbf{K}_{\mathbf{f},\star} & \mathbf{K}_{\star,\star}\end{bmatrix}\right) . \tag{2.2.15}$$

Calculating the predictive distribution in this case is identical to the analysis in Section 2.2.1, yielding:

$$p(\mathbf{f}_\star|\mathbf{y}) = \mathcal{N}(\mathbf{f}_\star|\boldsymbol{\mu}_\star, \boldsymbol{\Sigma}_\star) , \tag{2.2.16}$$

where,

$$\boldsymbol{\mu}_\star = m(\mathbf{x}_\star) + \mathbf{K}_{\star,\mathbf{f}}(\mathbf{K}_{\mathbf{f},\mathbf{f}} + \sigma^2_{\text{noise}} I)^{-1}(\mathbf{y} - m(\mathbf{x})) , \tag{2.2.17}$$

$$\boldsymbol{\Sigma}_\star = \mathbf{K}_{\star,\star} - \mathbf{K}_{\star,\mathbf{f}}(\mathbf{K}_{\mathbf{f},\mathbf{f}} + \sigma^2_{\text{noise}} I)^{-1}\mathbf{K}_{\mathbf{f},\star} . \tag{2.2.18}$$

Observe that the variance is not changed and as in the zero-mean case, it does not depend on the observed function values.

2.3 Sparse Gaussian process approximation

To train Gaussian processes models we need to compute the predictive distribution given by (2.2.10). As we can see, both the predictive mean (2.2.11) and the variance (2.2.12) involve the inversion of a matrix of size $N \times N$:

$$(\mathbf{K_{f,f}} + \sigma^2_{noise} I)^{-1} \,,$$

which has a computational complexity of $\mathcal{O}(N^3)$. The training time of a GP model will scale cubically with the number of training points, making GPs not viable for large datasets.

We are interested in modifying the (joint) prior distribution in ways that will reduce the computational complexity of the predictive equations. Some sparse approximations rely on using an approximate form of the likelihood (2.2.7), like the work of Smola and Bartlett [2001] or Seeger et al. [2003]. In this work we will focus on the approximation proposed in [Snelson and Ghahramani, 2006] which is sometimes also called, the fully independent training conditional approximation (FITC).

We consider an additional set of (D-dimensional) pseudo-inputs, also called inducing points, of size $M < N$, $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^{M}$ and its corresponding function values $\mathbf{u} = (f(\mathbf{z}_1), \ldots, f(\mathbf{z}_M))^T$. We also set a GP prior on the inducing variables as we expect them to be distributed in a similar way to $\mathbf{f}$ or $\mathbf{f}_\star$:

$$p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K_{u,u}}) \,, \tag{2.3.1}$$

where $\mathbf{K_{u,u}} = k(\mathbf{Z}, \mathbf{Z})$ is the kernel function evaluated at every pair of inducing points locations. The new joint prior is $p(\mathbf{f}, \mathbf{f}_\star, \mathbf{u})$ and from the GP definition we can recover the original joint prior (2.2.9) by marginalizing the inducing points from the new joint prior:

$$p(\mathbf{f}, \mathbf{f}_\star) = \int p(\mathbf{f}, \mathbf{f}_\star|\mathbf{u}) p(\mathbf{u}) d\mathbf{u} \approx \int p(\mathbf{f}|\mathbf{u}) p(\mathbf{f}_\star|\mathbf{u}) p(\mathbf{u}) d\mathbf{u} \,, \tag{2.3.2}$$

Note that the right hand side of the expression is an approximation, because $\mathbf{f}$ and $\mathbf{f}_\star$ do not factorize. Both conditionals are given by:

$$p(\mathbf{f}|\mathbf{u}) = \mathcal{N}(\mathbf{K_{f,u}} {\mathbf{K_{u,u}}}^{-1} \mathbf{u}, \mathbf{K_{f,f}} - \mathbf{Q_{f,f}}) \,, \tag{2.3.3}$$

$$p(\mathbf{f}_\star|\mathbf{u}) = \mathcal{N}(\mathbf{K}_{\star,\mathbf{u}} {\mathbf{K_{u,u}}}^{-1} \mathbf{u}, \mathbf{K}_{\star,\star} - \mathbf{Q}_{\star,\star}) \,, \tag{2.3.4}$$

and the matrix $\mathbf{Q}$ is defined as:

$$\mathbf{Q_{a,b}} \triangleq \mathbf{K_{a,u}} {\mathbf{K_{u,u}}}^{-1} \mathbf{K_{u,b}} \,. \tag{2.3.5}$$

We approximate the joint prior $p(\mathbf{f}, \mathbf{f}_\star|\mathbf{u})$ by assuming that $\mathbf{f}_\star$ and $\mathbf{f}$ are conditionally independent given $\mathbf{u}$. The FITC approximation can be seen as an approximation to the training conditional $p(\mathbf{f}|\mathbf{u})$ in (2.3.2), keeping the original likelihood (2.2.7) and test conditional $p(\mathbf{f}_\star|\mathbf{u})$, [Quiñonero-Candela and Rasmussen, 2005]. FITC assumes that the approximate conditional factorizes as:

$$p(\mathbf{f}|\mathbf{u}) \approx q_{\text{FITC}}(\mathbf{f}|\mathbf{u}) = \prod_{i=1}^{N} p(f_i|\mathbf{u}) = \mathcal{N}(f_i|\mathbf{K_{f,u}} {\mathbf{K_{u,u}}}^{-1} \mathbf{u}, \operatorname{diag}[\mathbf{K_{f,f}} - \mathbf{Q_{f,f}}]) \,, \tag{2.3.6}$$

where $\operatorname{diag}[\mathbf{K_{f,f}} - \mathbf{Q_{f,f}}]$ is a diagonal matrix with the values that matches those of $\mathbf{K_{f,f}} - \mathbf{Q_{f,f}}$. Using the approximate training conditional (2.3.6) we can recover the joint prior:

$$p(\mathbf{f}, \mathbf{f}_\star) \approx q(\mathbf{f}, \mathbf{f}_\star) = \int q_{\text{FITC}}(\mathbf{f}|\mathbf{u}) p(\mathbf{f}_\star|\mathbf{u}) p(\mathbf{u}) d\mathbf{u} \,, \tag{2.3.7}$$

The joint prior after marginalizing the inducing points is given by:

$$q(\mathbf{f}, \mathbf{f}_\star) = \mathcal{N}\left(\mathbf{f}, \mathbf{f}_\star \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{Q_{f,f}} - \operatorname{diag}[\mathbf{K_{f,f}} - \mathbf{Q_{f,f}}] & \mathbf{Q}_{\star,\mathbf{f}} \\ \mathbf{Q}_{\star,\mathbf{f}} & \mathbf{K}_{\star,\star} \end{bmatrix}\right) \,. \tag{2.3.8}$$

Where the matrices $\mathbf{Q}_{\mathbf{a},\mathbf{b}}$ are given in (2.3.5). Conditioning using the Gaussian identities (see Appendix A) we arrive at the predictive distribution for the test points:

$$p(\mathbf{f}_\star|\mathbf{y}) = \mathcal{N}(\mathbf{f}_\star|\mathbf{K}_{\star,\mathbf{u}}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\boldsymbol{\Lambda}^{-1}\mathbf{y},\;\; \mathbf{K}_{\star,\star} - \mathbf{Q}_{\star,\star} + \mathbf{K}_{\star,\mathbf{u}}\boldsymbol{\Sigma}^{-1}\mathbf{K}_{\mathbf{u},\star})\,, \tag{2.3.9}$$

with

$$\boldsymbol{\Sigma} = (\mathbf{K}_{\mathbf{u},\mathbf{u}} + \mathbf{K}_{\mathbf{u},\mathbf{f}}\boldsymbol{\Lambda}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}})\,, \tag{2.3.10}$$

$$\boldsymbol{\Lambda} = \mathrm{diag}[\mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{Q}_{\mathbf{f},\mathbf{f}} + \sigma^2_{\mathrm{noise}}\mathbf{I}]\,. \tag{2.3.11}$$

Note that the computational cost now is driven by the matrix multiplication $\mathbf{K}_{\mathbf{u},\mathbf{f}}\boldsymbol{\Lambda}^{-1}\mathbf{K}_{\mathbf{f},\mathbf{u}}$ which is $\mathcal{O}(M^2N)$ [Snelson and Ghahramani, 2006] because the matrix $\boldsymbol{\Lambda}$ is diagonal. Some computations in (2.3.9) can be simplified when using the matrix inversion lemma [Higham, 1996] and the mean and variance for predictions in (2.3.9) can be obtained with cost $\mathcal{O}(M)$ and $\mathcal{O}(M^2)$ respectively.

2.3.1 Selection of the inducing points locations

In the previous section, we calculated the predictive distribution for a sparse Gaussian process using a set of inducing points that we assumed as given. For the task of learning the locations $\mathbf{Z}$, we can treat them as model parameters. As such, the optimal locations can be found by optimizing the marginal likelihood conditioned on the inducing points.

We can compute the marginal likelihood $q(\mathbf{y}|\mathbf{Z})$ from the likelihood in (2.2.7), the training conditional in (2.3.6) and the prior on the inducing points in (2.3.1):

$$q(\mathbf{y}|\mathbf{Z}) = \int\int p(\mathbf{y}|\mathbf{f})q(\mathbf{f}|\mathbf{u})p(\mathbf{u}|\mathbf{Z})d\mathbf{u}d\mathbf{f}\,. \tag{2.3.12}$$

As in the full GP case, the marginal likelihood is once again a Gaussian distribution. We can apply the logarithm to simplify the task of finding the maximum:

$$q(\mathbf{y}|\mathbf{Z}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{Q}_{\mathbf{f},\mathbf{f}} + \mathrm{diag}[\mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{Q}_{\mathbf{f},\mathbf{f}}] + \sigma^2_{noise}I)\,, \tag{2.3.13}$$

$$\log q(\mathbf{y}|\mathbf{Z}) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{Q}_{\mathbf{f},\mathbf{f}} + \Lambda| - \frac{1}{2}\mathbf{y}^T(\mathbf{Q}_{\mathbf{f},\mathbf{f}} + \Lambda)^{-1}\mathbf{y}\,. \tag{2.3.14}$$

Note that as in Section 2.2, the rest of the model parameters can also be tuned by optimizing (2.3.13) using, for example, gradient ascent [Snelson and Ghahramani, 2006]. Figure 2.3.1 shows an example of regression using the FITC approximation. After maximizing the log marginal likelihood it can be seen that the locations of the inducing points are spread over the space of the inputs $\mathbf{x}$.

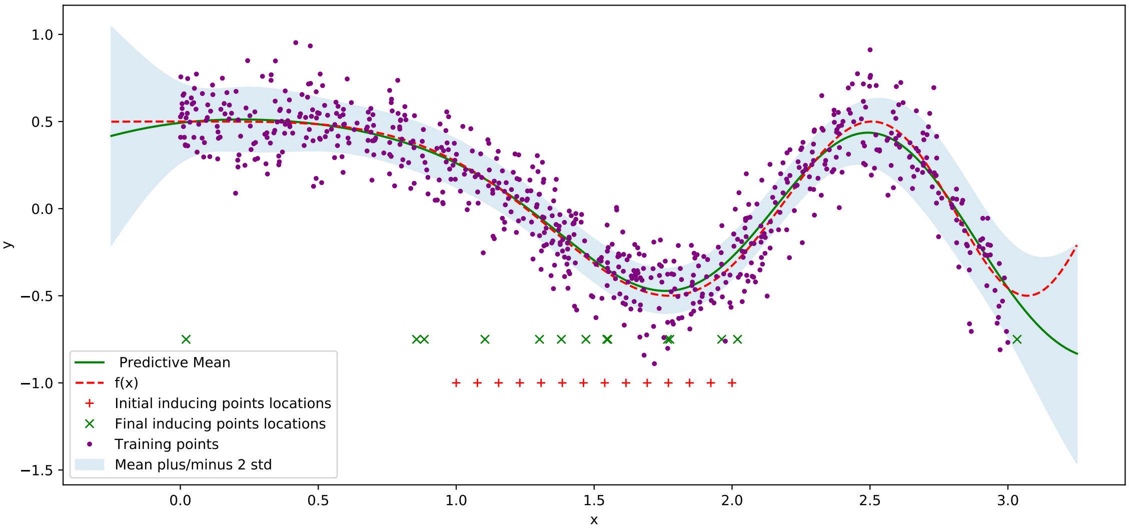


Figure 2.3.1: GP regression using the FITC approximation, with starting inducing points in red and inducing points locations after maximizing marginal likelihood in green. Note that the y axis values of the inducing points don't represent anything as we are only interested in their locations (x axis).

2.4 Summary

We have already seen that GPs present a lot of advantages over other models. GPs have fewer parameters to optimize that can be learned via evidence maximization, thus leading to models that are less prone to overfitting. It is possible to encode the knowledge or high-level properties of the problem easily by modifying the covariance function. Finally, they provide an uncertainty measure of the predictions. This usability and flexibility make them powerful models that have been successfully used in many areas like dynamical systems [Ko and Fox, 2009] or optimization [Pope et al., 2018]. However, GPs are limited by the kernel, and the global smoothness assumption made by widely used simple kernels such as RBF, can drastically reduce the performance in problems that present discontinuities in the data [Pope et al., 2018]. Some authors from the literature propose using less smooth but more practical kernel functions like the ARD Matérn kernel [Snoek et al., 2012]. Other approaches to model non-stationary data may include carefully combining kernels to create complex covariance functions [Duvenaud et al., 2013] [Rasmussen and Williams, 2006] or using constrained models [Cornford et al., 1998], see for example Figure 2.4.1.

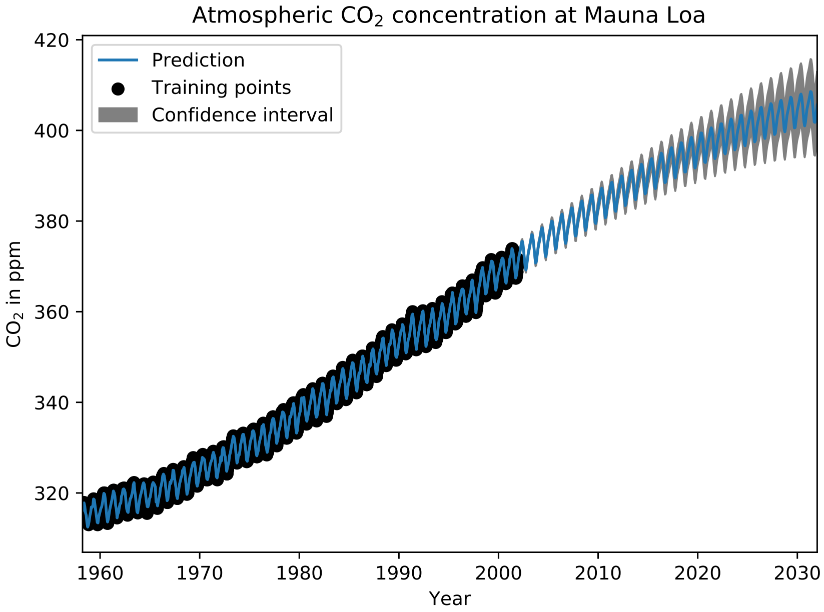


Figure 2.4.1: Gaussian process regression on Mauna Loa CO2 data. This dataset is non-stationary as its mean is increasing with time. Using a GP in this problem requires a covariance function composed of several different kernels (RBF, periodic kernel, rational quadratic, etc.). One drawback of this approach is the increased computational complexity or choosing the right kernels without prior information from the problem. The code to generate this figure has been extracted from [Pedregosa et al., 2011] and [Rasmussen and Williams, 2006].

Chapter 3

Approximate Inference

We have seen that Bayesian models have numerous advantages over other supervised learning methods, but sometimes exact inference in Bayesian models is not possible or it is computationally infeasible. Consider a model with latent parameters θ and observed data $\mathcal{D}$. From Bayes' theorem, we have that the posterior distribution of the parameters, given the data $p(\theta|\mathcal{D})$ is equal (up to a constant), to the product of the likelihood $p(\mathcal{D}|\theta)$ times the prior distribution $p(\theta)$ of the parameters:

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)d\theta}\,. \tag{3.0.1}$$

This posterior distribution can be used for making predictions about new unobserved data. When the likelihood is a conjugate distribution to the prior, the posterior can be calculated in closed form and the posterior and prior are then called conjugate probability distributions. If the likelihood lies in the exponential family then a conjugate prior exists [Gelman et al., 1995].

However, in real life applications, we may have to deal with distributions belonging to different families. Sometimes, calculating the normalization constant in the denominator of (3.0.1) is intractable so approximate inference or numerical integration needs to be used. The techniques that we are going to cover in this chapter focus on replacing the exact posterior distribution $p(\theta|\mathcal{D})$ by an approximation $q(\theta|\mathcal{D})$, which can be computed analytically and takes similar values to the exact distribution. The model evidence $p(\mathcal{D})$ is also useful for model comparison and some techniques also approximate this quantity.

3.1 Variational Inference

Variational inference is a technique that allows us to approximate an exact posterior distribution $p(\theta|\mathcal{D})$ with a new distribution $q(\theta)$ belonging to some tractable family. It also transforms an inference problem into an optimization one.

We could try to directly minimize the Kullback-Leibler divergence $\mathrm{KL}(q||p)$ which is a measure of similarity between two distributions given by:

$$\mathrm{KL}(q||p) = \int q(\theta) \ln \frac{q(\theta)}{p(\theta|\mathcal{D})} d\theta\,. \tag{3.1.1}$$

Note that the KL divergence is also valid in the case of distributions of discrete random variables, by substituting the integrals with summations. The KL divergence also satisfies the following properties:

$$\mathrm{KL}(q||p) \geq 0\,, \tag{3.1.2}$$

$$\mathrm{KL}(q||p) = 0 \iff q = p\,. \tag{3.1.3}$$

Another property is that the KL divergence is not symmetric, that is:

$$\mathrm{KL}(q||p) \neq \mathrm{KL}(p||q)\,.$$

The direction of the KL divergence determines where differences in the distributions are more important (either low or high values areas) [Wainwright and Jordan, 2008]. However, direct minimization of either quantities is not possible:

- The direct $\mathrm{KL}(p||q)$ requires evaluating the expectation $\mathbb{E}_{p(\theta|\mathcal{D})}[\ln \frac{q(\theta)}{p(\theta|\mathcal{D})}]$ and, in our case, the distribution $p(\theta|\mathcal{D})$ will correspond to an intractable posterior for which we will not be able to evaluate the required expectation.
- The reverse $\mathrm{KL}(q||p)$ will require to evaluate the expectation with respect to $q(\theta)$ which will be tractable; as we have chosen $q(\theta)$ to be from a distribution family that we can work with. However, when working with the true posterior distribution $p(\theta|\mathcal{D})$ we will still be required to evaluate the normalization constant $Z = p(\mathcal{D})$, given by the denominator in (3.0.1), which again is intractable.

Variational inference takes the second approach and tries to minimize the reverse KL divergence $\mathrm{KL}(q||p)$ indirectly. The main idea is that as our approximation $q(\theta)$ gets closer to the true posterior $p(\theta|\mathcal{D})$, the KL divergence will be reduced. First we can decompose the log probability of the evidence as follows:

$$\ln p(\mathcal{D}) = \ln \int p(\theta, \mathcal{D}) d\theta \tag{3.1.4}$$

$$= \ln \int p(\theta, \mathcal{D}) \frac{q(\theta)}{q(\theta)} d\theta \tag{3.1.5}$$

$$= \ln \left(\mathbb{E}_{q(\theta)} \left[\frac{p(\theta, \mathcal{D})}{q(\theta)} \right] \right)\,. \tag{3.1.6}$$

Because the log function is logarithmically concave, we can apply Jensen's inequality $f(\mathbb{E}[x]) \geq \mathbb{E}[f(x)]$ in (3.1.6):

$$\ln \left(\mathbb{E}_{q(\theta)} \left[\frac{p(\theta, \mathcal{D})}{q(\theta)} \right] \right) \geq \mathbb{E}_{q(\theta)} \left[\ln \frac{p(\theta, \mathcal{D})}{q(\theta)} \right]\,. \tag{3.1.7}$$

Finally, expanding the right hand side of (3.1.7) yields:

$$\mathcal{L}(q) = \mathbb{E}_{q(\theta)} \left[\ln \frac{p(\theta, \mathcal{D})}{q(\theta)} \right] = \mathbb{E}_{q(\theta)}[\ln p(\theta, \mathcal{D})] - \mathbb{E}_{q(\theta)}[\ln q(\theta)]\,. \tag{3.1.8}$$

The final term in (3.1.8) can also be expressed as the entropy of the distribution $q(\theta)$. It is easy to see then that $\ln p(\mathcal{D}) \geq \mathcal{L}(q)$. In other words, the functional $\mathcal{L}(q)$ acts as a lower bound of the evidence. In the literature $\mathcal{L}(q)$ is referred as the variational lower bound.

By using the definition of the KL divergence in (3.1.1) and the chain rule for probability $p(\theta, \mathcal{D}) = p(\theta|\mathcal{D})p(\mathcal{D})$, it is possible to decompose the KL divergence as:

$$\mathrm{KL}(q||p) = -\int q(\theta) \ln \frac{p(\theta|\mathcal{D})}{q(\theta)} d\theta \tag{3.1.9}$$

$$= -\int q(\theta) \frac{p(\theta, \mathcal{D})}{q(\theta)} d\theta + \ln p(\mathcal{D}) \int q(\theta) d\theta \tag{3.1.10}$$

$$= -\mathcal{L}(q) + \ln p(\mathcal{D})\,. \tag{3.1.11}$$

Where in (3.1.10) we have used that $\int q(\theta)d\theta = 1$ and introduced a probabilistic one $p(\mathcal{D})/p(\mathcal{D})$ to convert the conditional probability into the joint distribution of the variables $p(\theta, \mathcal{D})$. We can now rearrange (3.1.11):

$$\mathcal{L}(q) = -\mathrm{KL}(q||p) + \ln p(\mathcal{D}) \leq \ln p(\mathcal{D})\,. \tag{3.1.12}$$

Because the KL divergence is always ≥ 0, we have once again the property that $\mathcal{L}(q)$ acts as a lower bound of the evidence. Furthermore, we can see that the lower bound will be equal to the model evidence if and only if the KL divergence is zero, that is, when our approximate posterior is equal to the true posterior. The result of this derivation is that we can easily see that, maximizing the lower bound with respect to the distribution $q(\theta)$ will minimize the KL divergence between the exact distribution and our approximation (see Figure 3.1.1). By doing this we are assuming that $q(\theta)$ has some free parameters that we can tune, which are sometimes called "variational parameters".

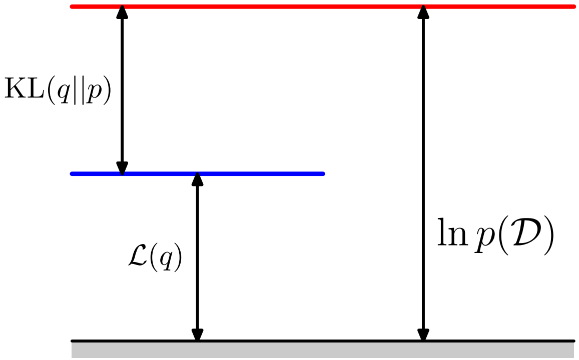


Figure 3.1.1: Visualization of the variational inference lower bound. Because the log likelihood $p(\mathcal{D})$ has a constant value for any distribution q and the KL divergence is always positive, we can see that maximizing the lower bound is equivalent to minimizing the KL divergence. Figure extracted from [Bishop, 2006].

3.2 Expectation Propagation

The expectation propagation (EP) algorithm is an extension of the assumed density method (ADF): a technique for computing approximate posterior distributions [Minka, 2001b]. The ADF algorithm (and also, EP) try to approximately minimize the KL divergence $\mathrm{KL}(p||q)$ between an exact distribution $p(\theta|\mathcal{D})$ and an approximate one $q(\theta)$, which belongs to a tractable family of distributions, while the variational inference algorithm seen in Section 3.1 minimizes the reverse KL divergence $\mathrm{KL}(q||p)$. In the ADF algorithm, updates are made sequentially so the final solution depends on the order in which the updates are made. Expectation propagation (EP) solves this problem by doing the updates repeatedly in multiple passes.

3.2.1 Expectation Propagation algorithm

In the EP framework, we assume that the exact posterior distribution can be expressed as a product of factors of the N independent observations [Bishop, 2006]:

$$p(\theta|\mathcal{D}) = \frac{1}{p(\mathcal{D})} p(\theta) \prod_{i=1}^{N} p(y_i|\theta)\,. \tag{3.2.1}$$

This expression can also be written as a function of the parameters:

$$p(\theta|\mathcal{D}) \propto p(\theta)\prod_{i=1}^{N} f_i(\theta) = \prod_{i=0}^{N} f_i(\theta)\,, \tag{3.2.2}$$

with $f_0(\theta) = p(\theta)$ and $f_i(\theta) = p(y_i|\theta)$. The model evidence $p(\mathcal{D})$ can be seen as a normalization constant and is given by:

$$p(\mathcal{D}) = \int p(\theta)\prod_{i=1}^{N} p(y_i|\theta)d\theta = \int \prod_{i=0}^{N} f_i(\theta)d\theta\,. \tag{3.2.3}$$

When working with Bayesian models we will usually be interested in calculating the posterior and the model evidence as well. EP assumes that our approximation will also be factorized in the same way: $q(\theta)$ will be the product of the approximate factors with the corresponding normalization constant Z:

$$q(\theta) = \frac{1}{Z}\prod_{i=0}^{N} \tilde{f}_i(\theta)\,. \tag{3.2.4}$$

The ideal value for the factor $\tilde{f}_i(\theta)$ would be the one that minimizes the KL divergence between the posterior in (3.2.2) and our approximation in (3.2.4) replacing one of the approximate factors with the exact one:

$$\min\nolimits_{\tilde{f}_i(\theta)} \mathrm{KL}(f_i(\theta)\prod_{j\neq i} \tilde{f}_j(\theta)\,||\,\prod_{i=0}^{N} \tilde{f}_i(\theta))\,. \tag{3.2.5}$$

Note that EP tries to minimize $\mathrm{KL}(p||q)$ (as opposed to variational inference, seen in Section 3.1), but direct minimization of that quantity is intractable because it requires to calculate the exact posterior. On the other hand minimizing a factor each time, as in (3.2.5) is a tractable problem. EP tries to approximate the exact posterior by sequentially computing an approximation $\tilde{f}_i(\theta)$ to every exact factor $f_i(\theta)$ and then constructing an approximate posterior with the approximations [Minka, 2001b]. This is achieved in a sequential procedure. For each factor we first remove it from the exact posterior to get a "cavity" term:

$$q^{\backslash i}(\theta) \propto \prod_{j\neq i} \tilde{f}_j(\theta) \propto \frac{q(\theta)}{\tilde{f}_i(\theta)}\,. \tag{3.2.6}$$

We now combine the cavity term with the corresponding exact factor in the so-called "tilted distribution":

$$\hat{p}_i(\theta) = \frac{1}{Z_i} f_i(\theta) q^{\backslash i}(\theta)\,. \tag{3.2.7}$$

Note that this expression appears on the left side of the KL divergence in (3.2.5). The normalization constant is given by $Z_i = \int f_i(\theta)q^{\backslash i}(\theta)d\theta$. We can now rewrite the KL divergence from (3.2.5) as:

$$\min\nolimits_{q(\theta)} \mathrm{KL}(\hat{p}_i(\theta)\,||\,q(\theta))\,. \tag{3.2.8}$$

The parameters for the approximate posterior $q_{\text{new}}(\theta)$ are obtained by minimizing the KL divergence. It is possible to show that this is equivalent to set the moments of the new approximate posterior equal to those of the tilted distribution, see [Seeger et al., 2003]. This is a tractable operation because usually, the tilted distribution belongs to a known distribution for which it is possible to compute the required moments [Bishop, 2006]. Finally the new (updated) factor can be obtained by removing the rest of the factors (cavity) from the new approximate posterior:

$$\tilde{f}_i(\theta) = Z_i \frac{q_{\text{new}}(\theta)}{q^{\setminus i}(\theta)} \,. \tag{3.2.9}$$

The model evidence can be approximated by substituting the exact factors in Eq. (3.2.3) with the approximate factors, using the normalization constant of $q(\theta)$ as an approximation:

$$p(\mathcal{D}) \approx \int \prod_{i=0}^{N} \tilde{f}_i(\theta) d\theta \,. \tag{3.2.10}$$

This procedure is repeated several times for all the factors. The EP algorithm is summarized in Algorithm 1.

```
Algorithm 1: EP algorithm
1. Initialize all approximate factors, also called "site factors" $\tilde{f}_i(\theta)$
2. Calculate the approximate posterior:
       $q(\theta) \propto \prod_{i=0}^{N} \tilde{f}_i(\theta) \,.$
while Not converged do
    3. Select a factor $\tilde{f}_i(\theta)$ to refine
    4. Calculate the "cavity distribution":
           $q^{\setminus i}(\theta) \propto \frac{q(\theta)}{\tilde{f}_i(\theta)} \,.$
    5. Calculate "tilted distribution":
           $\hat{p}_i(\theta) = \frac{1}{Z_i} f_i(\theta) q^{\setminus i}(\theta) \,.$
    6. Calculate $\hat{p}_i(\theta)$ moments set the parameters of the posterior $q_{\text{new}}(\theta)$ to those.
    7. Calculate the new factor:
           $\tilde{f}_i(\theta) = Z_i \frac{q_{\text{new}}(\theta)}{q^{\setminus i}(\theta)}$
end
8. Calculate the approximation to the model evidence with the normalization
   constant of $q(\theta)$:
       $p(\mathcal{D}) \approx \int \prod_{i=0}^{N} \tilde{f}_i(\theta) d\theta \,.$
```

3.2.2 Stochastic Expectation Propagation

One improvement to the EP algorithm that will be useful in further chapters of this thesis is the extension called Stochastic Expectation Propagation (SEP) [Li et al., 2015]. This method was created in an attempt to reduce the intensive memory requirements of the EP algorithm. Remember that as shown in Section 3.2 the updates for each of the approximate factors were done in a sequentially fashion and therefore each of these factors needs to be stored in memory. To illustrate this, consider a dataset comprising N i.i.d samples $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}$. From (3.2.4) we have that our approximation requires N different factors $\tilde{f}_i(\theta)$ that need to be stored in memory; in other words, the memory requirement is $\mathcal{O}(N)$.

The SEP approximation can be seen as a version of the EP algorithm in which each of the approximate factors $\tilde{f}_i(\theta)$ are forced to be equal (tied). This can be seen as each factor trying to capture the average effect of a likelihood term on the posterior [Li et al., 2015]. As in the EP case, we assume that our exact distribution can be expressed as the product of exact factors $f_i(\theta)$, but this time the approximate factors will all have the same form:

$$\underbrace{\tilde{f}(\theta)^N}_{\text{tied (SEP) factors}} \triangleq \underbrace{\prod_i^N \tilde{f}_i(\theta)}_{\text{original EP factors}} \approx \underbrace{\prod_i^N f_i(\theta)}_{\text{Factorized exact distribution}} . \tag{3.2.11}$$

With this modification the new approximation to the posterior becomes:

$$q(\theta) = \frac{1}{Z}\tilde{f}(\theta)^N . \tag{3.2.12}$$

With Z as the normalization constant assuring that the factors integrate to 1. The SEP approximation then follows the same steps as the EP algorithm, the cavity distribution is calculated by removing a single factor from (3.2.12):

$$q^{\setminus 1}(\theta) \propto \frac{q(\theta)}{\tilde{f}(\theta)} . \tag{3.2.13}$$

Using the new cavity distribution to calculate the tilted distribution with the exact factor we have:

$$\hat{p}_i(\theta) = \frac{1}{Z_i} f_i(\theta) q^{\setminus 1}(\theta) . \tag{3.2.14}$$

Note that, while the cavity will remain the same for all N factors, the tilted distribution will be different depending on which exact factor we include in it. By matching the moments of $q_{\text{new}}(\theta)$ to those of the tilted distribution $\hat{p}_i(\theta)$ we find a new temporal factor:

$$\tilde{f}_{\text{new}}(\theta) = Z_i \frac{q_{\text{new}}(\theta)}{q^{\setminus 1}(\theta)} . \tag{3.2.15}$$

This new factor just captures the effect of a single term from the likelihood, this is important when updating the approximate factor and therefore a damping technique must be used:

$$\tilde{f}(\theta) = \tilde{f}(\theta)^{1-\epsilon} \tilde{f}_{\text{new}}(\theta)^{\epsilon} , \tag{3.2.16}$$

here ϵ controls the dampening factor: usually $\epsilon = \frac{1}{N}$ is used, which can be seen as minimizing $\mathrm{KL}(\hat{p}_i(\theta)||q(\theta))$ (the same as in the EP case) [Li et al., 2015]. This improvement to EP effectively reduces the memory requirement to $\mathcal{O}(1)$ as the temporal factor in (3.2.15) does not need to be stored

3.3 Summary

In this section we have explored some techniques to approximate distributions based on the minimization of the Kullback-Leibler divergence between the approximation and the true distribution. VI minimizes the "reverse" KL divergence, KL($q||p$) while EP minimizes the "forward" KL divergence KL($p||q$). Due to the asymmetrical nature of this measure, the effects of each minimization are different. Minimizing the reverse form tends to produce approximations which avoid areas where the values of the true distribution are small, in other words, the reverse form tends to focus in areas near the modes of the true distribution. On the other hand, minimizing the forward KL divergence results in distributions that overestimate the variance of the true distribution; that is, it produces approximations that have high probability in areas where the true distribution has low probability [Bishop, 2006].

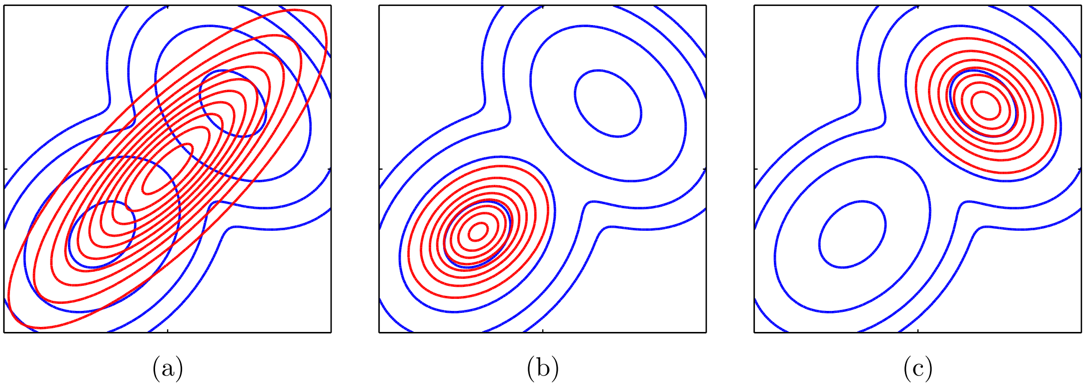

(a) (b) (c)

Figure 3.3.1: Comparison of the different minimizations of the KL divergence. A Gaussian mixture (in blue) is approximated by a single Gaussian (in red). (a) shows the result of minimizing the forward KL divergence. (b) and (c) shows the results of minimizing the reverse KL divergence with different initializations. See the text for more details. Figures extracted from [Bishop, 2006].

Figures 3.3.1a, 3.3.1b and 3.3.1c show a comparison of the different minimizations of the KL divergence. We can observe that, as expected, the minimization of the forward form produces an approximation which spreads more through the whole area of the true distribution, but gives high probability in the area between the two modes of the mixture, which is a low probability area in the true distribution. Minimizing the reverse form produces approximations that focus only on the modes of the true distribution while ignoring the other mode.

Chapter 4

Deep Gaussian Processes

Deep Gaussian process (DGP) are part of a family of models in machine learning called deep belief networks [Damianou and Lawrence, 2013]. These models are composed by multiple layers of hidden variables (nodes) [Hinton, 2009]. The term layer here is a simple abstraction to denote units or nodes in the network at the same depth level. These layers are connected in a feed-forward architecture and communication within nodes inside a layer is not possible. In DGPs the nodes represent GP functions feeding the output of each layer as the input to the next one. It has been shown that, under certain conditions, a GP is equivalent to a neural network with infinite units in the hidden layer [Neal, 1996b] and similarly, a DGP has a connection with deep neural networks [Neal, 1996b] [Lee et al., 2017]. This is illustrated in Figure 4.0.1.

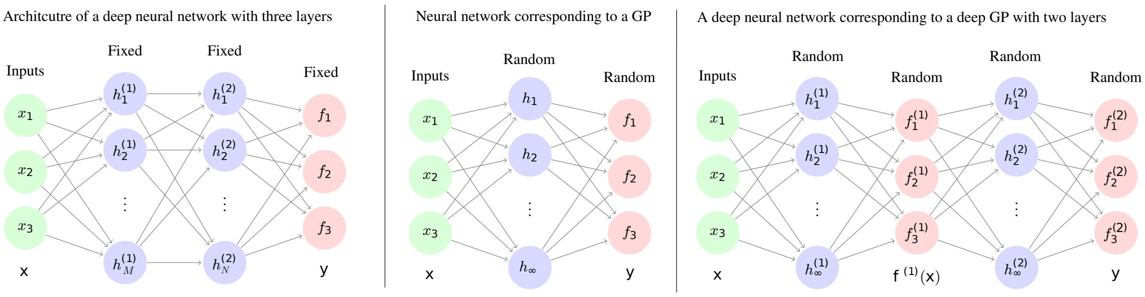

Architcutre of a deep neural network with three layers
Inputs
Fixed
Fixed
Fixed
x_1
x_2
x_3
x
$h_1^{(1)}$
$h_2^{(1)}$
$h_M^{(1)}$
$h_1^{(2)}$
$h_2^{(2)}$
$h_N^{(2)}$
f_1
f_2
f_3
y
Neural network corresponding to a GP
Inputs
Random
Random
x_1
x_2
x_3
x
h_1
h_2
h_∞
f_1
f_2
f_3
y
A deep neural network corresponding to a deep GP with two layers
Inputs
Random
Random
Random
Random
x_1
x_2
x_3
x
$h_1^{(1)}$
$h_2^{(1)}$
$h_\infty^{(1)}$
$f_1^{(1)}$
$f_2^{(1)}$
$f_3^{(1)}$
$f^{(1)}(x)$
$h_1^{(2)}$
$h_2^{(2)}$
$h_\infty^{(2)}$
$f_1^{(2)}$
$f_2^{(2)}$
$f_3^{(2)}$
y

Figure 4.0.1: A standard deep neural network (NN) architecture is shown on the left. Compare that with the single hidden layer NN corresponding to a shallow GP (middle) and the deep NN with two hidden layers corresponding to a DGP (right). A GP is equivalent to a NN with infinite units in the hidden layers where the output of said layers are random variables (as expected by the GP definition). Images are based in the work from Duvenaud et al. [2014].

In deep neural networks the mapping between layers is implemented as a parametric function of its inputs while in DGPs these functions are given by a GP. This allows DGPs to inherit some of the properties that make GP models useful such as nonparametric modeling and uncertainty estimates [Cutajar et al., 2017] [Bui et al., 2016]. DGPs allow us to model complex problems in which single layer GP models may struggle. Some of the limitations of GP models were explained in Section 2.4. With this definition, as a hierarchical composition of GP functions, taking samples from a DGP is straightforward and it can be done via forward sampling (see Figure 4.0.2), but as we will see, Bayesian inference in this model is analytically intractable.

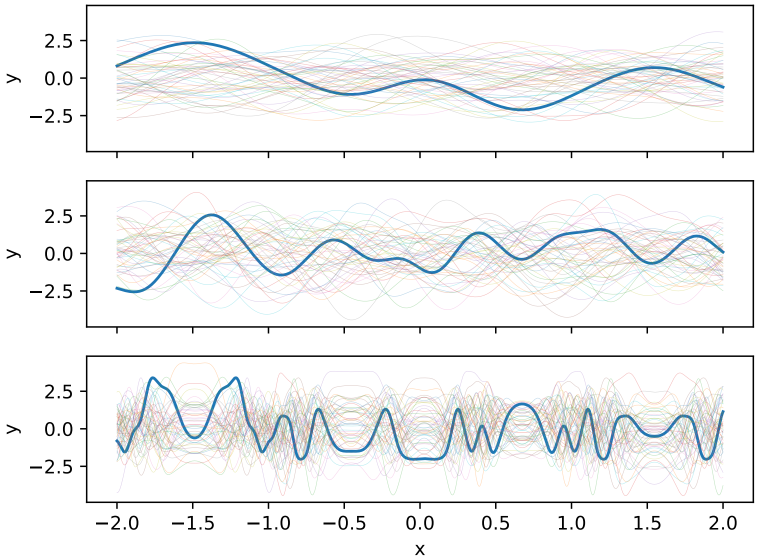


Figure 4.0.2: Samples taken from a DGP model with two layers using RBF kernel for both layers. The two plots at the top correspond to samples from the GP priors on each layer respectively. The figure at the bottom represents samples taken from the composition of both GPs, that is; the output of the first layers is feed as the input to the second layer. One sample in each figure is highlighted to improve readability. As we can see in the bottom picture, the DGP samples are complex functions in which the lengthscales do not remain constant trough the whole x axes.

4.1 The DGP model

For a given set of N observations $\mathbf{y} = (y_1, \ldots, y_N)^T$ and D-dimensional locations $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^T$, the corresponding DGP model representation with L layers has $\{\mathbf{H}_l\}_{l=1}^{L-1}$ as the output for each of the hidden layers. The number of columns in $\mathbf{H}_l$ is given by the number of nodes in the layer l, it is also called the "dimension of the layer" and can be written as D_l. More formally:

$$\mathbf{H}^l = \begin{bmatrix} h^l_{1,1} & \cdots & h^l_{1,D_l} \\ \vdots & \ddots & \vdots \\ h^l_{N,1} & \cdots & h^l_{N,D_l} \end{bmatrix}, \tag{4.1.1}$$

with:

$$h^l_{n,i} = f^{l,i}(\mathbf{h}^{l-1}_n). \tag{4.1.2}$$

Where $f(\cdot)$ is a latent function with a prior given by a GP. However, to lighten the notation, we will ignore the hidden variables dimension and abbreviate $\mathbf{H}^l$ as $\mathbf{h}^l$ and $f^{l,i}(\cdot)$ as $f^l(\cdot)$ for the functions in each layer. We now set a zero mean GP prior for each layer $p(f^l|\theta^l)$. In practice, for layers with multiples nodes, the prior is an independent GP function for each node inside each layer. This formulation corresponds to the case in which there is a single unit in the hidden layers. We assume some i.i.d Gaussian noise σ^2_l that can be parameterized at the output of each layer. The DGP model for the prior and the hidden variables can then be written as:

$$p(f^l|\theta^l) = \mathcal{GP}(f^l|\mathbf{0}, \mathbf{K}^l)\,, \quad l = 1, \dots, L\,, \tag{4.1.3}$$

$$p(\mathbf{h}^l|f^l, \mathbf{h}^{l-1}, \sigma_l^2) = \prod_{n=1}^{N} \mathcal{N}(h_n^l|f_l(h_n^{l-1}), \sigma_l^2)\,, \quad h_n^0 = \mathbf{x}_n\,. \tag{4.1.4}$$

Note that $\mathbf{K}^l$ represents the kernel matrix between the given inputs to the layer l, hence $\mathbf{K}^l = k(\mathbf{h}^{l-1}, \mathbf{h}^{l-1})$, and for layers with $l > 1$ the inputs will no longer be deterministic; therefore, the corresponding outputs will not follow a normal distribution. When $L = 1$ this model correspond to the shallow GP model from Chapter 2. Finally, the conditional probability of the given targets at the output layer is:

$$p(\mathbf{y}|f^L, \mathbf{h}^{L-1}, \sigma_L^2) = \prod_{n=1}^{N} \mathcal{N}(y_n|f_L(h_n^{L-1}), \sigma_L^2)\,. \tag{4.1.5}$$

An example of a two-layer model with one hidden layer and one output layer for a 2-D problem can be seen on Figure 4.1.1. The number of nodes at the output layer (D_L) will be equal to the dimension of the observations y_n for a regression problem, or the number of classes for a classification problem. However, in this thesis we will focus on regression problems with only one target variable. Therefore, at the output layer we will always consider a single output GP node.

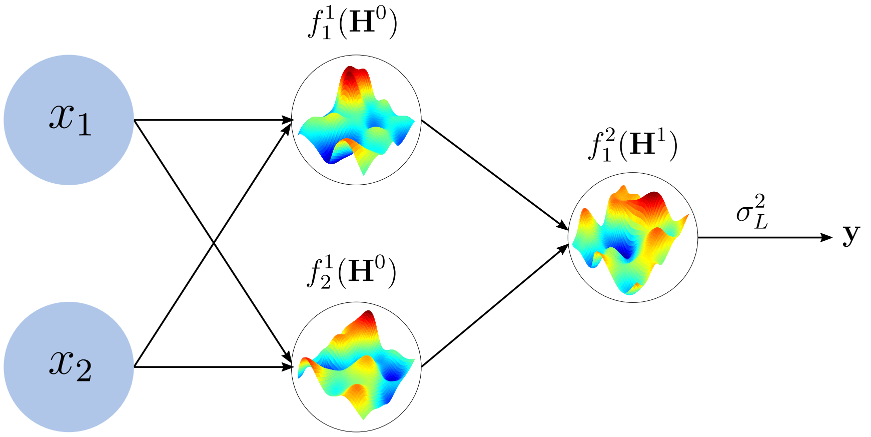


Figure 4.1.1: DGP example for a model with two layers $L = 2$. The nodes on each layer show the corresponding GP prior from Eq. (4.1.3). The 2-dimensional training points (on the left) are feed to the first layer (hidden) of the network, with $D_1 = 2$, the output at each layer is contaminated with some Gaussian noise (4.1.4).

4.1.1 Deep sparse GPs

As in the shallow GP model it is possible to incorporate the FITC sparse technique shown in Section 2.3 in the DGP model to reduce the computational complexity of the full DGP model. We define a set of M inducing points locations for a given layer l as $\mathbf{z}^{l-1} = (\mathbf{z}_1^{l-1}, \dots, \mathbf{z}_M^{l-1})^T$ with their corresponding values $\mathbf{u}^l = (f^l(z_1^{l-1}), \dots, f^l(z_M^{l-1}))^T$. Once again, we are omitting dimensions in our notation. Finally, we place GP priors on the inducing points of each layer. The final (sparse) DGP model can be written as [Bui et al., 2016]:

$$p(\mathbf{u}^l|\theta^l) = \mathcal{N}(\mathbf{u}^l|\mathbf{0}, \mathbf{K}^{-1}_{\mathbf{u}^l,\mathbf{u}^l}), \quad l = 1, \dots, L. \tag{4.1.6}$$

$$p(\mathbf{h}^l|\mathbf{u}^l, \mathbf{h}^{l-1}, \sigma_l^2) = \prod_{n=1}^{N} \mathcal{N}(h_n^l|\mathbf{A}_n^l\mathbf{u}^l,\ \mathbf{K}_{h_n^l,h_n^l} - \mathbf{Q}_n^l), \tag{4.1.7}$$

$$p(\mathbf{y}|\mathbf{u}^L, \mathbf{h}^{L-1}, \sigma_L^2) = \prod_{n=1}^{N} \mathcal{N}(y_n|\mathbf{A}_n^L\mathbf{u}^L,\ \mathbf{K}_{h_n^L,h_n^L} - \mathbf{Q}_n^l). \tag{4.1.8}$$

Here the covariance matrices $\mathbf{K}$ have as subindex their corresponding outputs; for example, $\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l}$ denotes the covariance matrix for the inducing points $\mathbf{u}^l$ which takes as argument its locations $\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l} = k(\mathbf{z}^{l-1}, \mathbf{z}^{l-1})$. The covariance matrix for the node in a layer $\mathbf{K}_{\mathbf{h}^l,\mathbf{h}^l}$ will be constructed with the output of the previous layer $\mathbf{K}_{\mathbf{h}^l,\mathbf{h}^l} = k(\mathbf{h}^{l-1}, \mathbf{h}^{l-1})$. We have also defined the matrices $\mathbf{A}$ and $\mathbf{Q}$ as:

$$\mathbf{A}_n^l \triangleq \mathbf{K}_{h_n^l,\mathbf{u}^l}\mathbf{K}^{-1}_{\mathbf{u}^l,\mathbf{u}^l}, \tag{4.1.9}$$

$$\mathbf{Q}_n^l \triangleq \mathbf{K}_{h_n^l,\mathbf{u}^l}\mathbf{K}^{-1}_{\mathbf{u}^l,\mathbf{u}^l}\mathbf{K}_{\mathbf{u}^l,h_n^l} + \sigma_l^2, \tag{4.1.10}$$

Figure 4.1.2 shows an example of the graphical model for the $L = 2$ case.

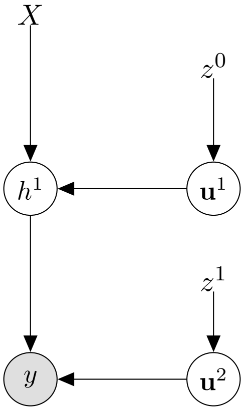


Figure 4.1.2: Graphical model for the Deep sparse GP with $L = 2$. Observe that now the hidden layers not only depend on the output of the layer about but also on the inducing points variables.

4.2 State of the art for inference in DGPs

We are now interested in doing inference in this model by marginalizing the latent variables; the inducing points $\{\mathbf{u}^l\}_{l=1}^L$ and the outputs of the hidden layers $\{\mathbf{h}^l\}_{l=1}^{L-1}$. This will allow us to make predictions about the test set with the posterior distribution and calculate the marginal likelihood $p(\mathbf{y})$ for hyperparameter tunning. However, as we will see, both of these quantities are intractable. As explained in Section 4.1.1, when $L = 1$, the model falls back to the single layer GP model explained in Chapter 2.

Consider now the DGP model when $L = 2$; a model with one hidden layer and one output layer. The joint distribution of the model is given by:

$$p(\mathbf{y}, \mathbf{h}^1, \{\mathbf{u}^l\}_{l=1}^2|\{\sigma_l^2, \theta^l\}_{l=1}^2, \mathbf{X}) = p(\mathbf{y}|\mathbf{u}^2, \mathbf{h}^1, \sigma_2^2)p(\mathbf{h}^1|\mathbf{u}^1, \mathbf{X}, \sigma_1^2)\prod_{l=1}^{2} p(\mathbf{u}^l|\theta^l). \tag{4.2.1}$$

For notational simplicity we will now group all the model parameters into:

$$\boldsymbol{\alpha} = \{\{\mathbf{z}^l\}_{l=0}^1, \{\theta^l, \sigma_l^2\}_{l=1}^2\}\,. \tag{4.2.2}$$

The marginal likelihood is then obtained by marginalizing $\{\mathbf{u}^l\}_{l=1}^2$ and the hidden variables $\mathbf{h}^1$ from (4.2.1):

$$p(\mathbf{y}|\boldsymbol{\alpha}, \mathbf{X}) = \int p(\mathbf{y}, \mathbf{h}^1, \{\mathbf{u}^l\}_{l=1}^2 | \boldsymbol{\alpha}, \mathbf{X}) d\mathbf{u}^1 d\mathbf{u}^2 d\mathbf{h}^1\,. \tag{4.2.3}$$

However some of the integrals in (4.2.3) are intractable, because they involve calculating covariance functions with respect to random variables [Damianou, 2015]. This can be seen by expanding the joint distribution inside the integral as in (4.2.1). From (4.1.7), we have that the corresponding distribution for the output in a layer $p(h^l|h^{l-1})$ requires the calculation of a non-linear kernel function that depends on the density for h^{l-1}.

Another quantity, needed to make predictions about test points will be the posterior distribution over the inducing points, which also requires the integrals in the model evidence (4.2.3) to be calculated:

$$p(\{\mathbf{u}^l\}_{l=1}^2 | \boldsymbol{\alpha}, \mathbf{X}, \mathbf{y}) = \frac{1}{p(\mathbf{y}|\boldsymbol{\alpha}, \mathbf{X})} \int p(\mathbf{y}, \mathbf{h}^1, \{\mathbf{u}^l\}_{l=1}^2 | \boldsymbol{\alpha}, \mathbf{X}) d\mathbf{h}^1\,. \tag{4.2.4}$$

This result can be extrapolated to the $L \geq 2$ case. For simplicity, from now on we will drop the layer dependency from the notation and abbreviate $\mathbf{u} = \{\mathbf{u}^l\}_{l=1}^L$ and $\mathbf{h} = \{\mathbf{h}^l\}_{l=1}^{L-1}$ for any arbitrary number of layers L. The (generalized) inducing points posterior becomes $p(\mathbf{u}|\boldsymbol{\alpha}, \mathbf{X}, \mathbf{y})$. In order to calculate the marginal likelihood and the posterior, approximate inference techniques are needed. In this section we will discuss about some of the state of the art techniques to approximate the quantities.

Some work in the DGP literature propose to use a general sampling algorithm [Vafa, 2016] to evaluate the log likelihood. The main difference with the methods explained in this section is that the sampling algorithm does not set any distribution over the inducing outputs (assumes the inducing outputs are fixed), and therefore they are included as model parameters. By doing this some of the regularization benefits are lost. They also propose to train the model via MAP (maximum a posterior estimation), but the authors did not compare the method with any other state of the art technique and they did not observe improvement when increasing the number of layers. Therefore, we have excluded this method in our experiments.

Another approach explained in [Cutajar et al., 2017] consists in approximating the kernel function $k(x, x')$ of a GP using random features vectors $\phi(\cdot)$. The kernel can be approximated as an inner product:

$$k(x, x') \approx \phi(x)^T \phi(x')\,.$$

The result is that the DGP model can be seen as a bayesian neural network (BNN). The output of a layer is given by $g(\mathbf{w}\mathbf{x} + \mathbf{b})$ with $g(\cdot)$ as the activation function, $\mathbf{w}$ is given by a probability distribution $p(\mathbf{w})$ and $\mathbf{b}$ is the bias term. For the RBF kernel used in all the DGP methods explained in this thesis, it corresponds to using a standard Gaussian prior over the weights $p(\mathbf{w}) = \mathcal{N}(\cdot, \cdot)$ and the sine and cosine functions as activation function. This allows to perform inference efficiently using techniques from BNNs methods. Because calculating the posterior distribution of the weights in a BNN requires approximate inference techniques the authors propose to use a factorised Gaussian (without dependencies) as an approximate distribution for the posterior of the weights which is not expected to yield optimal results.

4.2.1 Fully Factorized DGPs Using Variational Inference

The first method for doing inference in DGP models was shown in [Damianou and Lawrence, 2013] and [Damianou, 2015]. They propose to use a fully factorized variational posterior of the form:

$$p(\{\mathbf{h}^l, \mathbf{u}^l\}_{l=1}^L) \approx q(\{\mathbf{h}^l, \mathbf{u}^l\}_{l=1}^L) = \prod_{l=1}^{L} q(\mathbf{u}^l) q(\mathbf{h}^l)\,. \tag{4.2.5}$$

Observe that the approximation for the output of the hidden layers $q(\mathbf{h}^l)$ is removing the relations between layers that are present in the original model $p(\mathbf{h}^l|\mathbf{h}^{l-1})$. Because all the true distributions from the DGP model in Section 4.1.1 are Gaussian, the variational distributions in (4.2.5) are also taken to be Gaussian. With this choice, as seen in Section 3.1, the lower bound of Variational inference becomes [Damianou, 2015]:

$$\mathcal{L}(q) = \int\int q(\{\mathbf{h}, \mathbf{u}\}_{l=1}^L) \ln \frac{p(\mathbf{y}, \{\mathbf{h}^l, \mathbf{u}^l\}_{l=1}^L)}{q(\{\mathbf{h}, \mathbf{u}\}_{l=1}^L)} d\mathbf{u}^1, \cdots, d\mathbf{u}^L \quad d\mathbf{h}^1, \cdots, d\mathbf{h}^L\,. \tag{4.2.6}$$

Introducing the Gaussian distributions into the variational posterior we have [Damianou, 2015]:

$$q(\{\mathbf{H}, \mathbf{U}\}_{l=1}^L) = \prod_{l=1}^{L} \left[\prod_{j=1}^{D_l} \mathcal{N}([\mathbf{u}^l]_j \mid [\boldsymbol{\mu}^l]_j, [\boldsymbol{\Sigma}^l]_j) \prod_{i=1}^{n} \mathcal{N}([\mathbf{h}^l]_i \mid [\mathbf{m}^l]_i, [\mathbf{S}^l]_i) \right] , \tag{4.2.7}$$

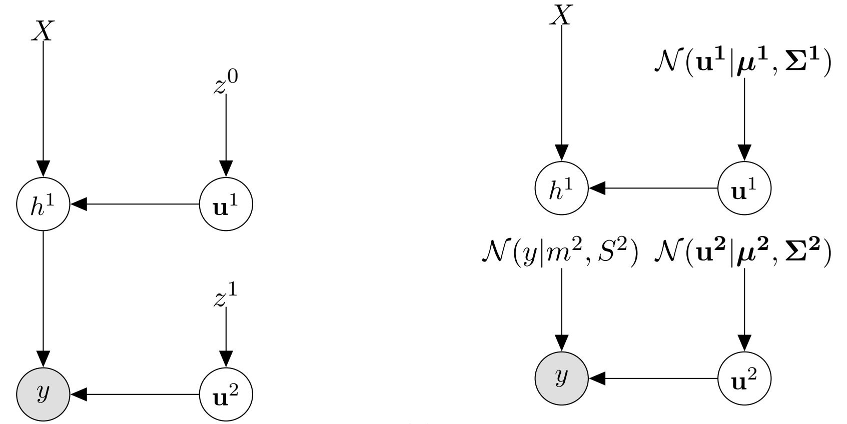
we have introduced again the dimensions on the notation to emphasize that the approximate factors in the variational posterior are fully factorized, more specifically: $\mathbf{H}^l$ is a $N \times D_l$ matrix with the outputs for the layer l and $\mathbf{U}^l$ is a $M \times D_l$ matrix with the inducing points values. The variational parameters for the distribution over the inducing points $\{[\boldsymbol{\mu}^l]_j, [\boldsymbol{\Sigma}^l]_j\}$ have sizes $(M \times 1)$ and $(M \times M)$ respectively, whereas the variational parameters for the outputs of the layers $\{[\mathbf{m}^l]_i, [\mathbf{S}^l]_i)\}$ have both sizes $(D_l \times 1)$ with $[\mathbf{S}^l]_i$ being a diagonal matrix due to the factorization. Finally, to find the optimal parameters for both the model and the variational posterior, it is possible to maximize the lower bound in (4.2.6). As shown in Figures 4.2.1a and 4.2.1a this choice of approximate posterior does not seem appropriate and consequently this model is not able to capture the complexity of the full DGP model since strong independence assumptions are enforced, which are not satisfied in practice.

4.2.2 DGPs for Regression Using Approximate Expectation Propagation

The approach proposed in [Bui et al., 2016] uses the Stochastic Expectation Propagation (SEP) modification to approximate the inducing points posterior while reducing the memory footprint of the EP algorithm. As seen in Section 3.2.2, SEP forces the factors to be tied; the approximate factors become effectively the same for all training points. We can write the approximation as:

$$p(\{\mathbf{u}^l\}_{l=1}^L|\mathbf{X}, \mathbf{y}) \approx q(\{\mathbf{u}^l\}_{l=1}^L) \propto \prod_{l=1}^{L} p(\mathbf{u}^l) g(\mathbf{u}^l)^N\,, \tag{4.2.8}$$

where $g(\mathbf{u}^l)$ is the approximate (tied) factor for the layer l. Remember that the posterior is proportional to the product of the prior times the likelihood. With the SEP approximation the factorized likelihood must be approximated with a single factor, hence the factor $g(\mathbf{u}^l)$


X
z^0
h^1
$\mathbf{u}^1$
z^1
y
$\mathbf{u}^2$
X
$\mathcal{N}(\mathbf{u^1}|\boldsymbol{\mu}^\mathbf{1}, \mathbf{\Sigma^1})$
h^1
$\mathbf{u}^1$
$\mathcal{N}(y|m^2, S^2)$
$\mathcal{N}(\mathbf{u^2}|\boldsymbol{\mu}^\mathbf{2}, \mathbf{\Sigma^2})$
y
$\mathbf{u}^2$

(a) Full sparse DGP model

(b) Fully factorized variational posterior DGP model.

Figure 4.2.1: Comparison of the full DGP model shown in Section 4.1.1 (left) with the model using a fully factorized variational posterior from Section 4.2.1. Note that this model ignores the important correlations between layers by making the input of a layer independent of the layer above.

can be thought as an average data factor that captures the average effect of a single likelihood term in the posterior [Bui et al., 2016]. With this modification, the cavity distribution becomes the same for all training points, reducing the memory requirement for the EP algorithm from $\mathcal{O}(NLM^2)$ to $\mathcal{O}(LM^2)$. The cavity distribution for a layer l is given by:

$$q^{\setminus l}(\mathbf{u}^l) \propto \frac{q(\mathbf{u}^l)}{g(\mathbf{u}^l)} \,. \tag{4.2.9}$$

An important difference with the SEP approximation proposed in [Li et al., 2015] is that they propose to optimize the EP energy function (the energy function is the EP estimate of the marginal likelihood) directly for the approximating factors and the hyperparameters, instead of doing the EP updates. Optimizing the full EP energy function, as an alternative to doing the EP updates, requires a double-loop algorithm [Minka, 2001a], but with the tied factors constraint this is no longer the case. The SEP approximation to the marginal likelihood is also simplified [Bui et al., 2016]:

$$\ln p(\mathbf{y}|\boldsymbol{\alpha}, \mathbf{X}) \approx \mathcal{F}(\boldsymbol{\alpha}) = \sum_{l=1}^{L} \left(\Phi(\theta_q^l) - \Phi(\theta_{\text{prior}}^l) + \sum_{n=1}^{N} \left[\ln \mathcal{Z}_n + \Phi(\theta^{\setminus l}) - \Phi(\theta_q^l) \right] \right) \tag{4.2.10}$$

$$= \sum_{l=1}^{L} \left[(1-N)\Phi(\theta_q^l) - \Phi(\theta_{\text{prior}}^l) + N\Phi(\theta^{\setminus l}) \right] + \sum_{n=1}^{N} \ln \mathcal{Z}_n \,, \tag{4.2.11}$$

with

$$\ln \mathcal{Z}_n = \ln \int p(y_n|\mathbf{u}^L, \mathbf{h}_n^{L-1}, \mathbf{x}_n) q^{\setminus L}(\mathbf{u}^L) \prod_{l=1}^{L-1} p(\mathbf{h}_n^l|\mathbf{h}_n^{l-1}, \mathbf{u}^l) q^{\setminus l}(\mathbf{u}^l) \; d\mathbf{u} d\mathbf{h} \,, \tag{4.2.12}$$

where the integrals in (4.2.12) are calculated for all the latent variables $d\{\mathbf{u}^l\}_{l=1}^{L}$, $d\{\mathbf{h}\}_{l=1}^{L-1}$ and $\theta_q^l, \theta^{\setminus l}, \theta_{\text{prior}}^l$ are the natural parameters of the corresponding distribution: either the approximate posterior $q(\mathbf{u}^l)$, the cavity $q^{\setminus l}(\mathbf{u}^l)$ or the prior $p(\mathbf{u}^l)$ respectively and $\Phi(\cdot)$ is the log-normalizer of the corresponding distribution.

Because optimizing the energy in (4.2.10) does not yield the same results as optimizing the full EP energy this new technique can be seen as an Approximate Expectation Propagation (AEP) method. With the AEP approximation and assuming that the tied factor $g(\mathbf{u})$ is a Gaussian, the natural parameters of the approximate posterior and the cavity can be calculated using the natural parameters θ of the tied factor [Seeger, 2005]:

$$\theta_q^l = \theta_{\text{prior}}^l + N\theta^l \,, \tag{4.2.13}$$

$$\theta^{\setminus l} = \theta_{\text{prior}}^l + (N-1)\theta^l \,. \tag{4.2.14}$$

The only not tractable quantity remaining to compute is $\ln \mathcal{Z}_n$. Once again, returning to the $L = 2$ case with one unit in each layer and expanding the notation we have, from (4.2.12), for a single multidimensional training point $\mathbf{x}_n$:

$$\mathcal{Z}_n = \int \int p(y|h^1, \mathbf{u}^2) q^{\setminus 2}(\mathbf{u}^2)\; d\mathbf{u}^2 \int p(h^1|\mathbf{x}_n, \mathbf{u}^1) q^{\setminus 1}(\mathbf{u}^1)\; d\mathbf{u}^1\; dh^1 \,. \tag{4.2.15}$$

In this case, because the approximate factor is assumed to be a Gaussian, the cavity distributions are also Gaussian:

$$q^{\setminus 1}(\mathbf{u}^1) = \mathcal{N}(\mathbf{u}^1|\boldsymbol{\mu}^{\setminus 1}, \boldsymbol{\Sigma}^{\setminus 1}) \,, \tag{4.2.16}$$

$$q^{\setminus 2}(\mathbf{u}^2) = \mathcal{N}(\mathbf{u}^2|\boldsymbol{\mu}^{\setminus 2}, \boldsymbol{\Sigma}^{\setminus 2}) \,. \tag{4.2.17}$$

Here, the superscript in the cavity $q^{\setminus 1}(\cdot)$ just denotes the corresponding layer and as seen earlier, is the same for all training points. As all the terms in (4.2.15) are Gaussian it is possible to analytically marginalize the inducing outputs $\mathbf{u}^1, \mathbf{u}^2$. The marginalization of $\mathbf{u}^1$ in the second integral from (4.2.15) is given by $q(h^1)$, while the marginalization of $\mathbf{u}^2$ results in the distribution $q(y|h^1)$, yielding [Bui et al., 2016]:

$$\mathcal{Z}_n = \int q(h^1) q(y|h^1) dh^1 = \int \mathcal{N}(h^1|m_1, v_1)\mathcal{N}(y|h^1 \mid m_{2|h^1}, v_{2|h^1}) dh^1 \tag{4.2.18}$$

$$\text{with} \quad q(h^1) = \mathcal{N}(h^1|m_1, v_1) \,, \tag{4.2.19}$$

$$m_1 = \mathbf{K}_{h^1,\mathbf{u}^1}\mathbf{K}_{\mathbf{u}^1,\mathbf{u}^1}^{-1}\boldsymbol{\mu}^{\setminus 1} \,,$$

$$v_1 = \sigma_1^2 + \mathrm{K}_{h^1,h^1} - \mathbf{K}_{h^1,\mathbf{u}^1}\mathbf{K}_{\mathbf{u}^1,\mathbf{u}^1}^{-1}\mathbf{K}_{\mathbf{u}^1,h^1} + \mathbf{K}_{h^1,\mathbf{u}^1}\mathbf{K}_{\mathbf{u}^1,\mathbf{u}^1}^{-1}\boldsymbol{\Sigma}^{\setminus 1}\mathbf{K}_{\mathbf{u}^1,\mathbf{u}^1}^{-1}\mathbf{K}_{\mathbf{u}^1,h^1} \,.$$

$$\text{And} \quad q(y|h^1) = \mathcal{N}(y|h^1 \mid m_{2|h^1}, v_{2|h^1}) \,, \tag{4.2.20}$$

$$m_{2|h^1} = \mathbf{K}_{y,\mathbf{u}^2}\mathbf{K}_{\mathbf{u}^2,\mathbf{u}^2}^{-1}\boldsymbol{\mu}^{\setminus 2} \,,$$

$$v_{2|h^1} = \sigma_2^2 + K_{y,y} - \mathbf{K}_{y,\mathbf{u}^2}\mathbf{K}_{\mathbf{u}^2,\mathbf{u}^2}^{-1}\mathbf{K}_{\mathbf{u}^2,y} + \mathbf{K}_{y,\mathbf{u}^2}\mathbf{K}_{\mathbf{u}^2,\mathbf{u}^2}^{-1}\boldsymbol{\Sigma}^{\setminus 2}\mathbf{K}_{\mathbf{u}^2,\mathbf{u}^2}^{-1}\mathbf{K}_{\mathbf{u}^2,y} \,.$$

Because h^1 is the output of the first GP layer, it is a normally distributed (random) variable with deterministic inputs and it can be calculated in closed form. The integral in (4.2.18) requires the calculation of a difficult integral of the function $q(y|h^1)$ in which h^1 follows the distribution $q(h^1)$ (is a random variable), which does not have an analytically closed form. Observe that if the input to the second layer was deterministic, as in the first layer case, the output could be calculated in a similar way to $q(h^1)$, which has a deterministic input $\mathbf{X}$. Although the integral in (4.2.18) cannot be calculated, it can be approximated with a Gaussian by applying the results from [Agathe Girard and Murray-Smith, 2003] and the law of total expectation and variance [Weiss et al., 2006]:

$$\mathcal{Z}_n \approx \mathcal{N}(y|m_2, v_2) \,, \tag{4.2.21}$$

$$m_2 = \mathbb{E}_{q(h^1)}\left[\mathbb{E}_{q(y|h^1)}\left[y|h^1\right]\right] = \mathbb{E}_{q(h^1)}\left[m_2|h^1\right] \,, \tag{4.2.22}$$

$$\begin{aligned} v_2 &= \mathbb{E}_{q(h^1)}\left[\mathrm{var}_{q(y|h^1)}\left[y|h^1\right]\right] + \mathrm{var}_{q(h^1)}\left[\mathbb{E}_{q(y|h^1)}\left[y|h^1\right]\right] \\ &= \mathbb{E}_{q(h^1)}\left[v_2|h^1\right] + \mathrm{var}_{q(h^1)}\left[m_2|h^1\right] \,. \end{aligned} \tag{4.2.23}$$

Expanding the equations above we have:

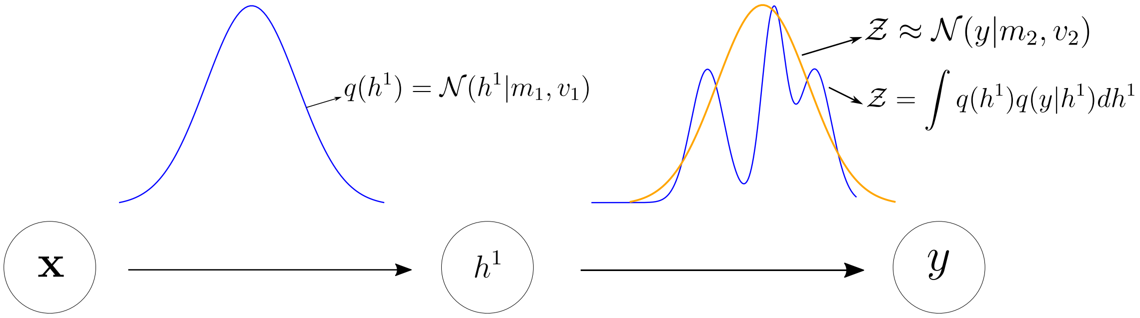
$$m_2 = \mathbb{E}_{q(h^1)}\left[\mathbf{K}_{y,\mathbf{u}^2}\right]\mathbf{K}^{-1}_{\mathbf{u}^2,\mathbf{u}^2}\boldsymbol{\mu}^{\setminus 2}\,, \tag{4.2.24}$$

$$v_2 = \sigma_2^2 + \mathbb{E}_{q(h^1)}\left[K_{y,y}\right] + \mathrm{tr}(\mathbf{A}\,\mathbb{E}_{q(h^1)}\left[\mathbf{K}_{\mathbf{u}^2,y}\mathbf{K}_{y,\mathbf{u}^2}\right]) - m_2^2\,, \tag{4.2.25}$$

with:

$$\mathbf{A} = \mathbf{K}^{-1}_{\mathbf{u}^2,\mathbf{u}^2}(\boldsymbol{\Sigma}^{\setminus 2} + \boldsymbol{\mu}^{\setminus 2}(\boldsymbol{\mu}^{\setminus 2})^T)\mathbf{K}^{-1}_{\mathbf{u}^2,\mathbf{u}^2} - \mathbf{K}^{-1}_{\mathbf{u}^2,\mathbf{u}^2}\,. \tag{4.2.26}$$

Here $\mathrm{tr}(\cdot)$ denotes the trace of a matrix. The expected values for the kernels in (4.2.24) and (4.2.25) can be calculated in closed form for some common kernels like the RBF kernel. An example of the procedure for calculating $\ln \mathcal{Z}$ in the DGP-AEP model can be seen in Figure 4.2.2.



$q(h^1) = \mathcal{N}(h^1|m_1, v_1)$
$\mathcal{Z} \approx \mathcal{N}(y|m_2, v_2)$
$\mathcal{Z} = \int q(h^1)q(y|h^1)dh^1$
$\mathbf{x}$
h^1
y

Figure 4.2.2: Procedure for computing $\ln \mathcal{Z}$ for a single point in a DGP model with 2 layers. At the first layer, $q(h^1)$ is given by a Gaussian distribution (top graph in blue) so it can be calculated in closed form. At the second layer the true distribution for $\ln \mathcal{Z}$ (in blue) is no longer a Gaussian and its given by (4.2.15). The distribution can be approximated by a Gaussian with the same moments (in orange) as explained in (4.2.21).

We are now interested in calculating the gradient $\partial \ln \mathcal{Z}/\partial \boldsymbol{\alpha}$ to find the best values for the hyperparameters. A forward and backward propagation step through the network is required in a similar fashion to the neural networks backpropagation algorithm [Bui et al., 2016]. We can first calculate $\mathcal{Z}$ with the equations above by propagating data through the network. We can then calculate the required gradients with respect to each of the parameters with any automatic differentiation library, to take steps in the negative direction of the gradient. Repeating this procedure until the parameters converge will yield the optimal values for the hyperparameters in the DGP-AEP model, as well as the parameters for the approximations. The memory and computational footprint of this procedure can furthermore be reduced by using minibatches.

The predictive distribution for the test inputs $\mathbf{x}^\star$ can be calculated in a similar way; by propagating the data through the network. The output, as in (4.2.21) can be approximated by a normal distribution, by exchanging the training inputs with the test inputs $\mathbf{x}^\star$ when calculating the corresponding kernel matrices $\mathbf{K}_{\mathbf{h}^1,\mathbf{u}^1}$ and using the approximate posterior instead of the cavity distribution:

$$\mathbf{K}_{\mathbf{h}^1,\mathbf{u}^1} \triangleq k(\mathbf{x}^\star, \mathbf{z}^1)\,, \tag{4.2.27}$$

$$\mathbf{K}_{\mathbf{h}^1,\mathbf{h}^1} \triangleq k(\mathbf{x}^\star, \mathbf{x}^\star)\,. \tag{4.2.28}$$

The predictive mean and variance (for $L \geq 2$) is given by:

$$p(\mathbf{y}^\star|\mathbf{x}^\star, \mathbf{X}, \mathbf{Y}, \boldsymbol{\alpha}) \approx \int p(\mathbf{y}^\star|\mathbf{h}^{L-1}, \mathbf{u}^L)q(\mathbf{u}^L) \prod_{l=1}^{L-1} p(\mathbf{h}^l|\mathbf{h}^{l-1}, \mathbf{u}^l)q(\mathbf{u}^l) \quad d\{\mathbf{u}^l\}_{l=1}^{L} d\{\mathbf{h}\}_{l=1}^{L-1}\,, \tag{4.2.29}$$

where $q(\mathbf{u}^l) = \mathcal{N}(\mathbf{u}^l|\boldsymbol{\mu}, \boldsymbol{\sigma})$ is the approximate posterior with its corresponding parameters. The input to the DGP is the test input $\mathbf{h}^0 \triangleq \mathbf{x}^\star$. The integral in (4.2.29) can be approximated as in (4.2.18):

$$p(\mathbf{y}^\star|\mathbf{x}^\star, \mathbf{X}, \mathbf{Y}, \boldsymbol{\alpha}) \approx \mathcal{N}(\mathbf{y}^\star|\hat{\mathbf{m}}, \hat{\mathbf{V}})\,, \tag{4.2.30}$$

$$\hat{\mathbf{m}} = \mathbb{E}_{q(\mathbf{h}^{L-1})}\left[\mathbf{K}_{\mathbf{y}^\star,\mathbf{u}^L}\right] \mathbf{K}^{-1}_{\mathbf{u}^L,\mathbf{u}^L} \boldsymbol{\mu}^L\,, \tag{4.2.31}$$

$$\hat{\mathbf{V}} = \sigma_2^2 \mathbf{I} + \mathbb{E}_{q(\mathbf{h}^{L-1})}\left[\mathbf{K}_{\mathbf{y}^\star,\mathbf{y}^\star}\right] + \operatorname{tr}(\hat{\mathbf{A}}\,\mathbb{E}_{q(\mathbf{h}^{L-1})}\left[\mathbf{K}_{\mathbf{u}^L,\mathbf{y}^\star}\mathbf{K}_{\mathbf{y}^\star,\mathbf{u}^L}\right]) - \hat{\mathbf{m}}^2\,. \tag{4.2.32}$$

with

$$\hat{\mathbf{A}} = \mathbf{K}^{-1}_{\mathbf{u}^L,\mathbf{u}^L}(\boldsymbol{\Sigma}^L + \boldsymbol{\mu}^L(\boldsymbol{\mu}^L)^T)\mathbf{K}^{-1}_{\mathbf{u}^L,\mathbf{u}^L} - \mathbf{K}^{-1}_{\mathbf{u}^L,\mathbf{u}^L}\,. \tag{4.2.33}$$

4.2.3 Doubly Stochastic Variational Inference for DGPs

The work in [Salimbeni and Deisenroth, 2017] proposes to approximate the posterior distribution with a variational posterior that maintains the correlations between layers while simplifying the correlations within each layer node using variational inference (see Section 3.1). The joint distribution for a model with L layers, omitting dimensions in the notation, is given by:

$$p(\mathbf{y}, \{\mathbf{h}^l, \mathbf{u}^l\}_{l=1}^L) = \prod_{i=1}^{N} p(y_i|h_i^L) \prod_{l=1}^{L} p(\mathbf{h}^l|\mathbf{h}^{l-1}, \mathbf{u}^l)p(\mathbf{u}^l)\,, \quad \text{where} \quad \mathbf{h}^0 = \mathbf{X}\,. \tag{4.2.34}$$

As explained in the model from Section 4.1.1: $\mathbf{y}$ are the observed training values, the output values for a layer l with the corresponding M inducing values $\mathbf{u}^l$ are given by $\mathbf{h}^l$. The first term $p(\mathbf{y}|\mathbf{h}^L)$ is the (Gaussian) likelihood of the model $\mathcal{N}(\mathbf{y}|\mathbf{h}^L, \sigma_L^2\mathbf{I})$, the hidden layers conditionals $p(\mathbf{h}^l|\mathbf{h}^{l-1}, \mathbf{u}^l)$ are the same as in (4.1.7) with the exception that the noise between layers is absorbed into the kernel function:

$$k_{\text{noisy}} = k(\mathbf{x}_i, \mathbf{x}_j) + \delta_{i,j}\sigma_l^2\,, \tag{4.2.35}$$

where $\delta_{i,j}$ is the Kronecker delta. Finally $p(\mathbf{u}^l)$ is the inducing points prior for each of the layers, as in (4.1.6). All terms in (4.2.34) are Gaussian, however, as we have already seen in Section 4.1.1, inference in this model is intractable due to the complex dependencies between the layers.

The proposed variational posterior in [Salimbeni and Deisenroth, 2017] maintains the conditional on the inducing points $p(\mathbf{h}^l|\mathbf{h}^{l-1}, \mathbf{u}^l)$ and assumes that the posterior distribution of the inducing points $\mathbf{u}^l$ factorizes across layers. The approximate variational posterior has the following form:

$$q(\{\mathbf{h}^l, \mathbf{u}^l\}_{l=1}^L) = \prod_{l=1}^{L} p(\mathbf{h}^l|\mathbf{h}^{l-1}, \mathbf{u}^l)q(\mathbf{u}^l)\,, \tag{4.2.36}$$

As opposed to the technique explained in Section 4.2.1, the proposed posterior maintains the hierarchical layer structure of the model (as the output of a layer $\mathbf{h}^l$ depends on the output of the layer $\mathbf{h}^{l-1}$). Keeping the true conditional on the inducing points this method allows to capture some of the complex relations of the sparse DGP model. Furthermore, the inducing points posterior has been approximated by a Gaussian distribution, which takes the simple form:

$$q(\mathbf{u}^l) = \mathcal{N}(\mathbf{u}^l|\mathbf{m}^l, \mathbf{S}^l)\,, \quad l = 1, \dots, L\,. \tag{4.2.37}$$

This choice helps simplifying the correlations inside each of the layers of the model by removing dependencies in the parameters of the Gaussian. Because both terms in the approximate posterior (4.2.36) are Gaussian, the inducing variables can be marginalized analytically as follows:

$$\begin{aligned} q(\{\mathbf{h}^l\}_{l=1}^L) &= \int \cdots \int q(\{\mathbf{h}^l, \mathbf{u}^l\}_{l=1}^L) \quad d\mathbf{u}^1, \cdots, d\mathbf{u}^L \qquad (4.2.38)\\ &= \prod_{l=1}^L q(\mathbf{h}^l|\mathbf{h}^{l-1}) = \prod_{l=1}^L \mathcal{N}(\mathbf{h}^l|\tilde{\boldsymbol{\mu}}^l, \tilde{\boldsymbol{\Sigma}}^l)\,. \end{aligned}$$

where the parameters of the Gaussian can be calculated using the law of total expectation and variance [Weiss et al., 2006] and have the following form for the i-th training example:

$$[\tilde{\boldsymbol{\mu}}^l]_i = \mathbb{E}_{q(\mathbf{u}^l)}\left[p(\mathbf{h}^l|\mathbf{h}^{l-1}, \mathbf{u}^l)\right] = \mathbf{K}_{h_i^l,\mathbf{u}^l}\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l}^{-1}\mathbf{m}^l\,, \tag{4.2.39}$$

$$\begin{aligned} [\tilde{\boldsymbol{\Sigma}}^l]_i &= \mathbb{E}_{q(\mathbf{u}^l)}\left[\mathrm{var}_p\left(p(\mathbf{h}^l|\mathbf{h}^{l-1}, \mathbf{u}^l)\right)\right] + \mathrm{var}_{q(\mathbf{u}^l)}\left[\mathbb{E}_p\left(p(\mathbf{h}^l|\mathbf{h}^{l-1}, \mathbf{u}^l)\right)\right]\\ &= K_{h_i^l,h_i^l} - \mathbf{K}_{h_i^l,\mathbf{u}^l}\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l}^{-1}\mathbf{K}_{\mathbf{u}^l,h_i^l} + \mathbf{K}_{h_i^l,\mathbf{u}^l}\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l}^{-1}\mathbf{S}^l\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l}^{-1}\mathbf{K}_{\mathbf{u}^l,h_i^l}\,. \qquad (4.2.40) \end{aligned}$$

Here, for simplicity we are assuming that the mean for the priors is 0, and as in Section 4.1.1, $\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l}$ represents the kernel function between each pair of the corresponding inputs $\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l} = k(\mathbf{z}^{l-1}, \mathbf{z}^{l-1})$. One important property that can be extracted from (4.2.39) and (4.2.40) is that the marginal h_i^L (the output of the DGP for the i-th training example) depends only on the i-th output of the predecessor layers in a hierarchical manner. By marginalizing all outputs for the hidden layers $\{h_i^l\}_{l=1}^{L-1}$ from (4.2.38) we arrive at the distribution for the variational posterior for the DGP model:

$$q(h_i^L) = \int \prod_{l=1}^{L-1} q(h_i^l|h_i^{l-1}; \mathbf{m}^l, \mathbf{S}^l, \mathbf{z}^{l-1}) dh_i^l\,. \tag{4.2.41}$$

We have explicitly added the dependency on the variational parameters $\mathbf{m}^l, \mathbf{S}^l$ and inducing points locations $\mathbf{z}^{l-1}$ for each layer l.

The proposed posterior does not make inference tractable, but the already mentioned property makes sampling from it possible by propagating samples through the network using the "re-parameterization trick" [Rezende et al., 2014] [Kingma et al., 2015]. In particular, to draw a sample from $q(h_i^L)$ we first sample from the standard Gaussian distribution:

$$\epsilon_i^l \sim \mathcal{N}(0, 1)\,. \tag{4.2.42}$$

Then, for all the layers $l = 1\,, \dots\,, L - 1$ and using (4.2.39) and (4.2.40), we draw samples

$\hat{h}_i^l$ from the marginal $q(h_i^l|h_i^{l-1})$ as follows [Salimbeni and Deisenroth, 2017]:

$$\hat{h}_i^l \sim q(h_i^l|h_i^{l-1})\,, \tag{4.2.43}$$

$$\text{where}\quad \hat{h}_i^l = \mathbf{K}_{\hat{h}_i^l,\mathbf{u}^l}\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l}^{-1}\mathbf{m}^l \tag{4.2.44}$$

$$+\,\epsilon_i^l\sqrt{K_{\hat{h}_i^l,\hat{h}_i^l} - \mathbf{K}_{\hat{h}_i^l,\mathbf{u}^l}\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l}^{-1}(\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l}^{-1} - \mathbf{S}^l)\mathbf{K}_{\mathbf{u}^l,\mathbf{u}^l}^{-1}\mathbf{K}_{\mathbf{u}^l,\hat{h}_i^l}}\,,$$

$$\text{and}\quad \hat{h}_i^0 = \mathbf{x}_i\,. \tag{4.2.45}$$

Note that the first term (4.2.44) is the mean from (4.2.39) and the value inside the square root is just the variance from (4.2.40). From (3.1.8), we have that the lower bound of the marginal likelihood for the DGP model is given by:

$$\mathcal{L}_{\text{DGP}}(q) = \mathbb{E}_{q(\{\mathbf{h}^l,\mathbf{u}^l\}_{l=1}^L)}\left[\ln\frac{p(\mathbf{y},\{\mathbf{h}^l,\mathbf{u}^l\}_{l=1}^L)}{q(\{\mathbf{h}^l,\mathbf{u}^l\}_{l=1}^L)}\right] \tag{4.2.46}$$

$$= \mathbb{E}_{q(\{\mathbf{h}^l,\mathbf{u}^l\}_{l=1}^L)}\left[\ln\frac{\prod_{i=1}^N p(y_i|h_i^L)\prod_{l=1}^L \cancel{p(\mathbf{h}^l|\mathbf{h}^{l-1},\mathbf{u}^l)}p(\mathbf{u}^l)}{\prod_{l=1}^L \cancel{p(\mathbf{h}^l|\mathbf{h}^{l-1},\mathbf{u}^l)}q(\mathbf{u}^l)}\right] \tag{4.2.47}$$

$$= \int\left[\int q(\{\mathbf{h}^l,\mathbf{u}^l\}_{l=1}^L)d\{\mathbf{u}^l\}_{l=1}^L\ \ln\left(\prod_{i=1}^N p(y_i|h_i^L)\right)\right]\ d\{\mathbf{h}^l\}_{l=1}^L \tag{4.2.48}$$

$$+\int\left[\int q(\{\mathbf{h}^l,\mathbf{u}^l\}_{l=1}^L)d\{\mathbf{h}^l\}_{l=1}^L\ \ln\left(\frac{\prod_{l=1}^L p(\mathbf{u}^l)}{\prod_{l=1}^L q(\mathbf{u}^l)}\right)\right]\ d\{\mathbf{u}^l\}_{l=1}^L \tag{4.2.49}$$

$$= \int q(\{\mathbf{h}^l\}_{l=1}^L)\ln\left(\prod_{i=1}^N p(y_i|h_i^L)\right)\ d\{\mathbf{h}^l\}_{l=1}^L \tag{4.2.50}$$

$$+\int q(\{\mathbf{u}^l\}_{l=1}^L)\ln\left(\frac{\prod_{l=1}^L p(\mathbf{u}^l)}{\prod_{l=1}^L q(\mathbf{u}^l)}\right)\ d\{\mathbf{u}^l\}_{l=1}^L \tag{4.2.51}$$

$$= \sum_{i=1}^N \mathbb{E}_{q(h_i^L)}\left[\ln p(y_i|h_i^L)\right] - \sum_{l=1}^L \mathrm{KL}(q(\mathbf{u}^l)||p(\mathbf{u}^l))\,. \tag{4.2.52}$$

Here in (4.2.47) we have expanded both the joint distribution and the approximate posterior canceling some terms. In (4.2.48) we have applied the expected value definition and made some rearranging. After marginalizing variables from the inner integrals in the joint posterior for both terms, we see that the likelihood in (4.2.50) only depends on the output of the last layer $\mathbf{h}^L$ and we can express it as an expected value. Finally the second term (4.2.51) can be recognized as the Kullback-Leibler divergence between the approximate distribution of the inducing points and the exact prior on the inducing points.

The lower bound can be evaluated by taking samples with the procedure explained above for the variational DGP posterior from (4.2.41). Because the lower bound is a sum of independent terms it is possible to optimize it using minibatches and stochastic gradients, reducing the memory requirements and the overall computational cost. This enables the use of this model on very large datasets.

Finally, the prediction for a test point $\mathbf{x}_\star$ can be calculated by sampling from the variational posterior (4.2.41) changing the input locations $\mathbf{x}$ to the test location $\mathbf{x}_\star$ [Salimbeni and Deisenroth, 2017]:

$$q(h_\star^L) = \frac{1}{S}\sum_{s=1}^S q(h_\star^l|h_\star^{L-1};\mathbf{m}^L,\mathbf{S}^L,\mathbf{z}^{L-1})\,, \tag{4.2.53}$$

where S is the number of samples for the Monte Carlo estimate of the variational posterior.

4.3 Deep Gaussian Processes Using Expectation Propagation and Monte Carlo Methods

Our contribution (DGP-AEPMCM) follows the approach explained in [Bui et al., 2016] and approximates the inducing points posterior using the expectation propagation algorithm with the tied factors constraint (4.2.8). As in Section 4.2.2 the inducing points posterior is approximated with a distribution $q(\mathbf{u})$ using approximate expectation propagation:

$$p(\{\mathbf{u}^l\}_{l=1}^L|\mathbf{X},\mathbf{y}) \approx q(\{\mathbf{u}^l\}_{l=1}^L) \propto \prod_{l=1}^{L} p(\mathbf{u}^l)g(\mathbf{u}^l)^N\,, \tag{4.3.1}$$

with cavity:

$$q^{\backslash l}(\mathbf{u}^l) \propto \frac{q(\mathbf{u}^l)}{g(\mathbf{u}^l)}\,. \tag{4.3.2}$$

and keeping the original inducing points prior $p(\mathbf{u}^l)$ from (4.1.6). As in (4.2.10), the SEP approximation to the marginal like likelihood is given by:

$$\ln p(\mathbf{y}|\boldsymbol{\alpha},\mathbf{X}) \approx \mathcal{F}(\boldsymbol{\alpha}) = \sum_{l=1}^{L}\left(\Phi(\theta_q^l) - \Phi(\theta_{\text{prior}}^l) + \sum_{n=1}^{N}\left[\ln \mathcal{Z}_n + \Phi(\theta^{\backslash l}) - \Phi(\theta_q^l)\right]\right) \tag{4.3.3}$$

$$= \sum_{l=1}^{L}\left[(1-N)\Phi(\theta_q^l) - \Phi(\theta_{\text{prior}}^l) + N\Phi(\theta^{\backslash l})\right] + \sum_{n=1}^{N}\ln \mathcal{Z}_n\,, \tag{4.3.4}$$

with

$$\ln \mathcal{Z}_n = \ln \int p(y_n|\mathbf{u}^L,\mathbf{h}_n^{L-1},\mathbf{x}_n)q^{\backslash L}(\mathbf{u}^L)\prod_{l=1}^{L-1} p(\mathbf{h}_n^l|\mathbf{h}_n^{l-1},\mathbf{u}^l)q^{\backslash l}(\mathbf{u}^l)\; d\mathbf{u}d\mathbf{h}\,, \tag{4.3.5}$$

Again, $\theta_q^l\,, \theta^{\backslash l}\,, \theta_{\text{prior}}^l$ are the natural parameters for the approximate posterior $q(\mathbf{u}^l)$, the cavity $q^{\backslash l}(\mathbf{u}^l)$ or the prior $p(\mathbf{u}^l)$ respectively, which in our case are Gaussian distributions. $\Phi(\cdot)$ is the log partition function of the corresponding distribution. The cavity distribution is also a Gaussian of the form:

$$q^{\backslash l}(\mathbf{u}^l) \triangleq \mathcal{N}(\mathbf{u}^l|\boldsymbol{\mu}^{\backslash l},\boldsymbol{\Sigma}^{\backslash l})\,, \tag{4.3.6}$$

To calculate $\mathcal{Z}_n$ we proceed by first marginalizing the inducing points, as in Section 4.2.2. We set the number of layers $L = 2$ and one hidden unit in the layers for simplicity. For a single multidimensional training point $\mathbf{x}_n$ we have:

$$\mathcal{Z}_n = \int\int p(y|h^1,\mathbf{u}^2)q^{\backslash 2}(\mathbf{u}^2)\; d\mathbf{u}^2 \int p(h^1|\mathbf{x}_n,\mathbf{u}^1)q^{\backslash 1}(\mathbf{u}^1)\; d\mathbf{u}^1\; dh^1 \tag{4.3.7}$$

$$= \int q(h^1)q(y|h^1)dh^1 = \int \mathcal{N}(h^1|m_1,v_1)\mathcal{N}(y|h^1 \mid m_{2|h^1},v_{2|h^1})dh^1\,. \tag{4.3.8}$$

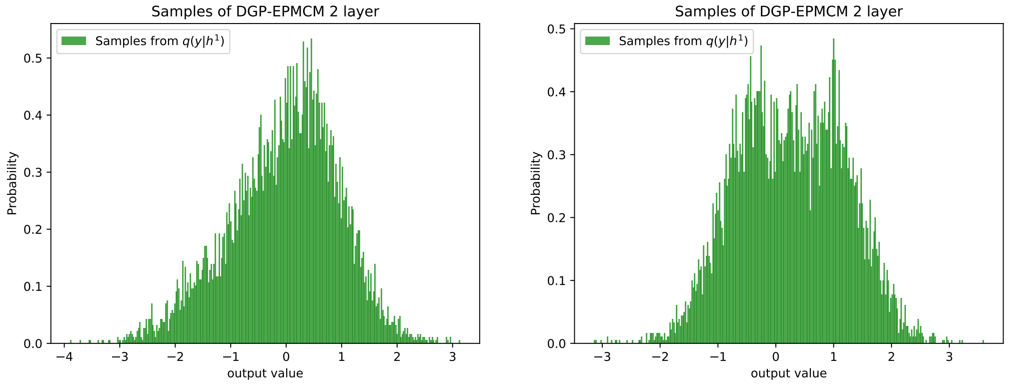
We can now extend to the more general case of a DGP with L layers:

$$\mathcal{Z}_n = \int q(y|h^{L-1})q(h^1|\mathbf{x}_n)\prod_{l=2}^{L-1} q(h^l|h^{l-1})\; d\{h\}_{l=1}^{L-1}\,. \tag{4.3.9}$$

By recognizing that $h^0 \triangleq \mathbf{x}_n$ and $h^L \triangleq y$, we can simplify (4.3.9) into:

$$\mathcal{Z}_n = \int \prod_{l=1}^{L} q(h^l|h^{l-1})\ d\{h\}_{l=1}^{L-1} = \int \prod_{l=1}^{L} \mathcal{N}(h^l|m_l, v_l)\ d\{h\}_{l=1}^{L-1}\,. \tag{4.3.10}$$

The integral above is intractable when $L > 1$ and, as shown in Figures 4.3.1a and 4.3.1b, it is a complex distribution with multiple modes, hence the Gaussian approximation proposed in [Bui et al., 2016] does not seem like a appropriate choice.



(a) (b)

Figure 4.3.1: 8000 samples taken from $q(y|h^1)$ in Eq.(4.3.8) for different inputs x, the plots show histograms of the probability of the output y.

Our approach uses a Monte Carlo method to estimate $\mathcal{Z}_n$. For a layer l, given a deterministic input $\hat{h}^{l-1}$, the distribution over the hidden variable for the output of the layer can be calculated analytically, as in (4.2.44), in the Doubly Stochastic Variational approach:

$$q(h^l|\hat{h}^{l-1}) = \mathcal{N}(h^l|m^l, v^l) \quad l = 1, \ldots, L\,, \quad \text{with } \hat{h}^0 = x_n\,,\ h^L = y\,, \tag{4.3.11}$$

$$m^l = \mathbf{K}_{\hat{h}^l,\mathbf{u}^l}\mathbf{K}^{-1}_{\mathbf{u}^l,\mathbf{u}^l}\boldsymbol{\mu}^{\setminus l}\,, \tag{4.3.12}$$

$$v^l = \sigma_l^2 + K_{\hat{h}^l,\hat{h}^l} - \mathbf{K}_{\hat{h}^l,\mathbf{u}^l}\mathbf{K}^{-1}_{\mathbf{u}^l,\mathbf{u}^l}\mathbf{K}_{\mathbf{u}^l,\hat{h}^l} + \mathbf{K}_{\hat{h}^l,\mathbf{u}^l}\mathbf{K}^{-1}_{\mathbf{u}^l,\mathbf{u}^l}\boldsymbol{\Sigma}^{\setminus l}\mathbf{K}_{\mathbf{u}^l,\hat{h}^l}\mathbf{K}_{\mathbf{u}^l,\hat{h}^l}\,. \tag{4.3.13}$$

Here $\boldsymbol{\Sigma}^{\setminus l}$ and $\boldsymbol{\mu}^{\setminus l}$ are the parameters of the cavity distribution from (4.3.6). The noise added at the output variance of each layer l is given by σ_l^2 and the kernel matrices can be calculated as:

$$K_{\hat{h}^l,\hat{h}^l} = k(\hat{h}^{l-1}, \hat{h}^{l-1})\,, \qquad \mathbf{K}_{\hat{h}^l,\mathbf{u}^l} = k(\hat{h}^{l-1}, \mathbf{u}^{l-1})\,, \tag{4.3.14}$$

$$\mathbf{K}^{-1}_{\mathbf{u}^l,\mathbf{u}^l} = k(\mathbf{u}^{l-1}, \mathbf{u}^{l-1})^{-1}\,, \qquad \mathbf{K}_{\mathbf{u}^l,\hat{h}^l} = \mathbf{K}^T_{\hat{h}^l,\mathbf{u}^l} = k(\mathbf{u}^{l-1}, \hat{h}^{l-1})\,. \tag{4.3.15}$$

We now can take a sample from $q(h^l|\hat{h}^{l-1})$ by sampling from standard univariate Gaussian distributions in a similar fashion to the process described in [Salimbeni and Deisenroth, 2017]. Given a sample ϵ from a standard Gaussian distribution:

$$\epsilon \sim \mathcal{N}(0, 1)\,. \tag{4.3.16}$$

We can sample from the distribution $q(h^l|\hat{h}^{l-1})$ for $l = 1, \ldots, L$ given that $\hat{h}^{l-1}$ is deterministic (either other samples or the input to the DGP) by changing the scale and location using (4.3.12) and (4.3.13):

$$\hat{h}^l \sim q(h^l|\hat{h}^{l-1}) \tag{4.3.17}$$

$$\text{where} \quad \hat{h}^l = m^l + \epsilon\sqrt{v^l}\,, \tag{4.3.18}$$

$$\hat{h}^0 = \mathbf{x}_n\,.$$

The final form for $\mathcal{Z}_n$ approximated with S samples is given by a Gaussian mixture:

$$\mathcal{Z}_n \approx \frac{1}{S}\sum_{s=1}^{S} q(y|\hat{h}_s^{L-1})\,. \tag{4.3.19}$$

Where $\hat{h}_s^{L-1}$ denotes the s-th sample taken from the corresponding distribution $q(\hat{h}_s^{L-1}|\hat{h}_s^{L-2})$ and can be calculated with the sampling technique explained above. Figure 4.3.2 shows an example of the procedure to propagate samples through the DGP network.

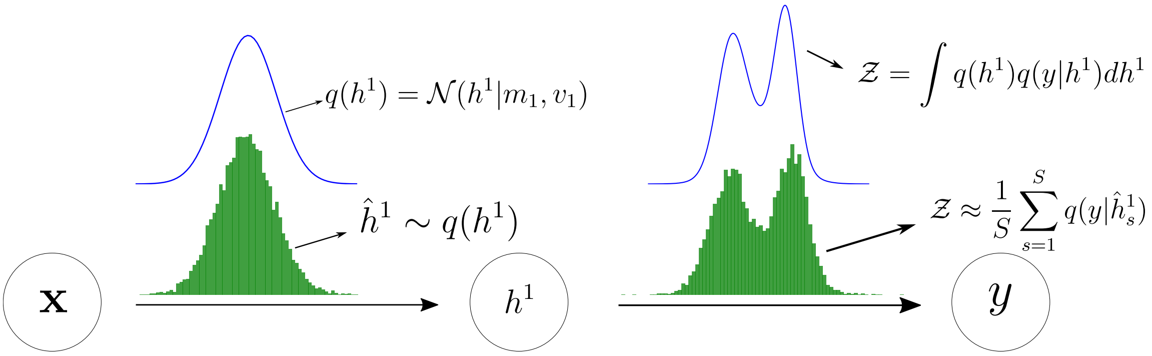

$q(h^1) = \mathcal{N}(h^1|m_1, v_1)$
$\mathcal{Z} = \int q(h^1)q(y|h^1)dh^1$
$\hat{h}^1 \sim q(h^1)$
$\mathcal{Z} \approx \frac{1}{S}\sum_{s=1}^{S} q(y|\hat{h}_s^1)$
$\mathbf{x}$
h^1
y

Figure 4.3.2: Procedure for computing $\ln \mathcal{Z}$ for a single point in a DGP model with 2 layers. At the first layer, $q(h^1)$ is given by a Gaussian distribution (top graph in blue) from which we sample from. At the second layer the true distribution for $\ln \mathcal{Z}$ (in blue) is no longer a Gaussian and its given by (4.3.10). Our proposed approach calculates and propagates samples through the network (in green) which is expected to make the model more flexible by being able to work with complex distributions.

As opposed to the procedure proposed in [Bui et al., 2016] this method can capture the complex dependencies between the DGP layers as shown in Figure 4.3.3. As in [Bui et al., 2016], our method is also suitable for stochastic gradient descent techniques such as minibatch training. The final form for the marginal likelihood approximation is given by:

$$\mathcal{F}(\boldsymbol{\alpha}) = \sum_{l=1}^{L}\Big[(1-N)\Phi(\theta_q^l) - \Phi(\theta_{\text{prior}}^l) + N\Phi(\theta^{\setminus l})\Big] + \frac{N}{|B|}\sum_{n=1}^{|B|}\ln \mathcal{Z}_b\,, \tag{4.3.20}$$

(4.3.21)

where $\boldsymbol{\alpha}$ include the model and AEP parameters to tune, $|B|$ is the chosen minibatch size and $\ln \mathcal{Z}_b$ can be calculated using (4.3.19) for each of the minibatches.

4.3.1 Predictive distribution

The predictive distribution for a single test point $\mathbf{x}^\star$ is also given by a mixture of Gaussians:

$$p(y^\star|\mathbf{x}^\star, \mathbf{X}, \mathbf{Y}, \boldsymbol{\alpha}) \approx q(y^\star|\hat{h}^{L-1}) = \frac{1}{S}\sum_{s=1}^{S}\mathcal{N}(y^\star|m_s^L, v_s^L)\,, \tag{4.3.22}$$

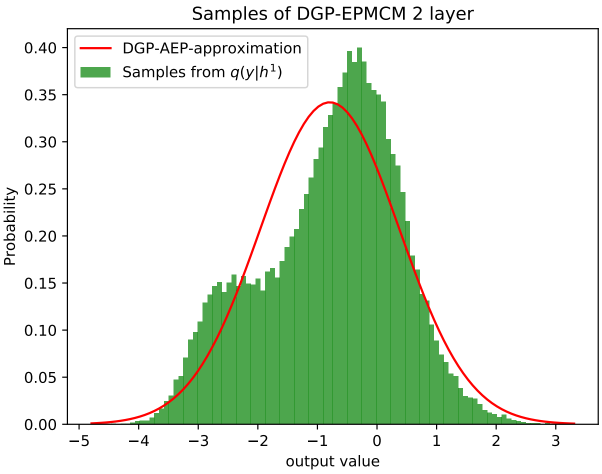


Figure 4.3.3: Comparison of the two methods explained in this thesis to calculate $\ln \mathcal{Z}_n$ for a DGP model with 2 layers. The DGP-AEP method proposed in [Bui et al., 2016] approximates the output distribution $q(y|h^1)$ with a Gaussian distribution (in red). The histogram of the samples from our method to calculate $\ln \mathcal{Z}$ are shown in green. We see that, in this case, the Gaussian approximation fails to capture the bimodality from distribution $q(y|h^1)$.

where the parameters of each of the components of the Gaussian mixture m_s^L and v_s^L can be calculated via sampling and have the following form for the s-th sample $\hat{h}_s^{L-1}$:

$$m_s^L = \mathbf{K}_{\hat{h}^L, \mathbf{u}_s^L} \mathbf{K}_{\mathbf{u}^L, \mathbf{u}^L}^{-1} \boldsymbol{\mu}^L , \tag{4.3.23}$$

$$v_s^L = \sigma_l^2 + K_{\hat{h}_s^L, \hat{h}_s^L} - \mathbf{K}_{\hat{h}^L, \mathbf{u}_s^L} \mathbf{K}_{\mathbf{u}^L, \mathbf{u}^L}^{-1} \mathbf{K}_{\mathbf{u}^L, \hat{h}_s^L} + \mathbf{K}_{\hat{h}^L, \mathbf{u}_s^L} \mathbf{K}_{\mathbf{u}^L, \mathbf{u}^L}^{-1} \mathbf{\Sigma}^L \mathbf{K}_{\mathbf{u}^L, \hat{h}_s^L} \mathbf{K}_{\mathbf{u}^L, \hat{h}_s^L} . \tag{4.3.24}$$

Here, the parameters of the cavity distribution from (4.3.12) and (4.3.13) have been substituted for the parameters of the posterior $q(\mathbf{u}^L) = \mathcal{N}(\mathbf{u}^L | \boldsymbol{\mu}^L, \mathbf{\Sigma}^L)$ Finally, the mean and variance of the Gaussian mixture can be calculated as:

$$\mathbb{E}_{q(y^\star | \hat{h}^{L-1})}[y^\star] = \frac{1}{S} \sum_{s=1}^{S} \mathbb{E}_q[\mathcal{N}(y^\star | m_s^L, v_s^L)] = \frac{1}{S} \sum_{s=1}^{S} m_s^L , \tag{4.3.25}$$

$$\mathrm{var}_q[y^\star] = \mathbb{E}_q[{y^\star}^2] - \mathbb{E}_q[y^\star]^2 = \left[\frac{1}{S} \sum_{s=1}^{S} {m_s^L}^2 + v_s^L \right] - \left[\frac{1}{S} \sum_{s=1}^{S} m_s^L \right]^2 \tag{4.3.26}$$

Figure 4.3.4 shows how the predictive distribution is able to explain how the data was generated with a multimodal distribution in areas where both values (−1 and 1) are possible for the step function problem. This behavior is explained and compared in more detail in Section 5.2.

4.3.2 Mean function

Experiments from [Duvenaud et al., 2014] and [Salimbeni and Deisenroth, 2017] show that as the number of layers increase in the DGP model, the prior functions generated by this model become highly non-injective by either becoming nearly flat or with extremely high variance. This can cause trouble when trying to recovering the original x values from the y

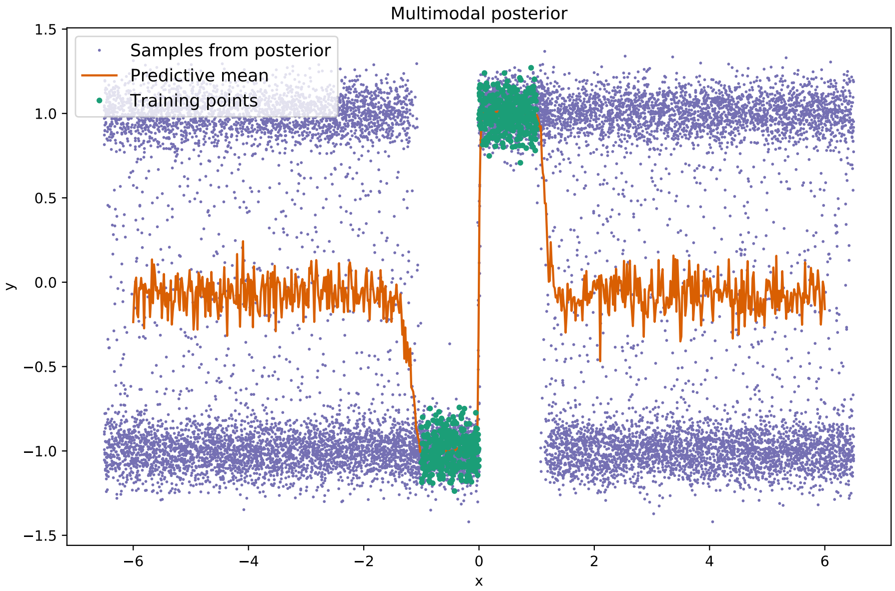


Figure 4.3.4: Predictive distribution for a DGP with 3 layers for the step function problem. The plot shows that the samples (in purple) taken outside of the training $\mathbf{x}$ locations (in green) are generated from a multimodal distribution.

values at the training phase. Figure 4.3.5 shows samples taken from the DGP model prior with 6 layers.

To solve this problem, the authors in [Salimbeni and Deisenroth, 2017] suggest adding a linear function to the mean in each of the hidden layers.

$$m(\mathbf{X}) = \mathbf{X}\mathbf{W}\,. \tag{4.3.27}$$

Here $\mathbf{X}$ is the input to the GP layer l, of size $N \times D^{l-1}$, with D^{l-1} being the dimension (the number of nodes) in the previous layer. For the case that the previous layers is the same dimension $D^{l-1} = D^l$ the identity matrix is used for $\mathbf{W}$. For the case where $D^{l-1} < D^l$ we use the first D^{l-1} rows of the identity matrix of size $D^l \times D^l$ for $\mathbf{W}$. Finally for the case where $D^{l-1} > D^l$ we use the first D^l values of the SVD decomposition of the original input to the network $\mathbf{X}$.

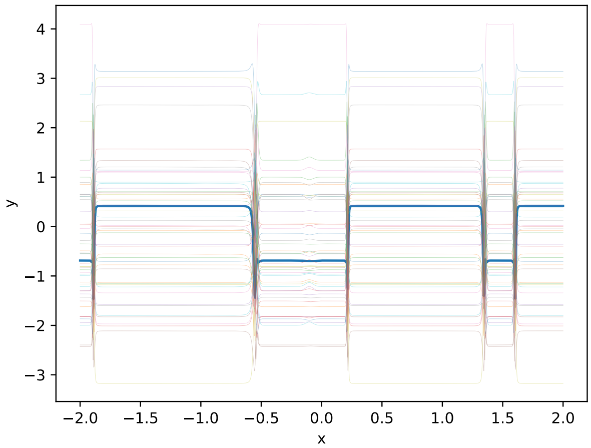


Figure 4.3.5: Samples taken from the DGP-AEPMCM model with 6 layers, it is possible to see that the functions generated by the DGP model are non-injective, making nearly impossible to recover the original x value from a given y value. Problems can arise during the training phase due to this pathology. One sample is highlighted to improve readability.

Chapter 5

Experiments

To run our experiments, we have chosen to implement the proposed method in Tensorflow [Abadi et al., 2015] using Sacred [Klaus Greff et al., 2017] to log all the results. We evaluate the performance of the proposed approach with three main experiments by comparing it with two of the state of the art methods for DGPs, including:

- DGP using variational inference (from [Salimbeni and Deisenroth, 2017]), explained in Section 4.2.3. Abbreviated as DGP-VI.
- DGP using Approximate Expectation propagation (from [Bui et al., 2016]), explained in Section 4.2.2. Abbreviated as DGP-AEP.
- The proposed method in this master thesis, DGP using Approximate Expectation propagation and Monte Carlo methods, explained in Section 4.3. Abbreviated as DGP-AEPMCM.

In the first experiment we compare the performance for eight UCI datasets evaluating root mean squared error (RMSE) and log-likelihood (LL); two of the most used metrics in the literature for Bayesian models. Table 5.0.1 shows a summary of all the UCI datasets used for this experiment. For the second experiment we compare the predictive distributions for the DGP-VI and DGP-AEPMCM models.

Table 5.0.1: Summary of the UCI datasets used in our experiments.

Dataset Name	N	D
Boston	506	13
Concrete	1030	8
Energy	768	8
Kin8	8191	8
Naval	11934	16
Power	9568	4
Protein	45730	9
Wine	1599	11

Finally, to demonstrate that our proposal can be used in Big data problems, we compare the performance for each of the three methods for two large scale datasets: Year (N = 515,345, D=90) and Airline (N = 2,082,007, D=8). We measure the elapsed training time, log-likelihood and RMSE scores for each method.

5.1 Regression problems

Following [Salimbeni and Deisenroth, 2017], we evaluate the root mean squared error (RMSE) and test log-likelihood (LL), on 8 regressions problems from the UCI repository. We use the same architecture for all the models: a minibatch size of 100, using Adam [Kingma and Ba, 2014], with all the parameters set to the default values except the learning rate that we set to 0.01. We train for 2000 epochs for the smaller datasets (Boston, Concrete, Energy, Kin8, Wine) and 500 for the bigger ones (Naval, Power, Protein). All the models use the RBF kernel and 100 inducing points. We use 20-fold cross validation, randomly selecting 10% of the data in each partition for test.

We compare models with 2, 3, 4 and 5 layers for all the methods. The dimension for the hidden layers is chosen to be the minimum between the dimension of the problem and 30 as in [Salimbeni and Deisenroth, 2017]. Table 5.1.1 shows the RMSE results and Table 5.1.2 shows the log-likelihood. Results for the DGP-AEP model for the protein dataset are not reported due to excessive computation time. Figures 5.1.1 and 5.1.2 show a graphical representation of the results in the tables.

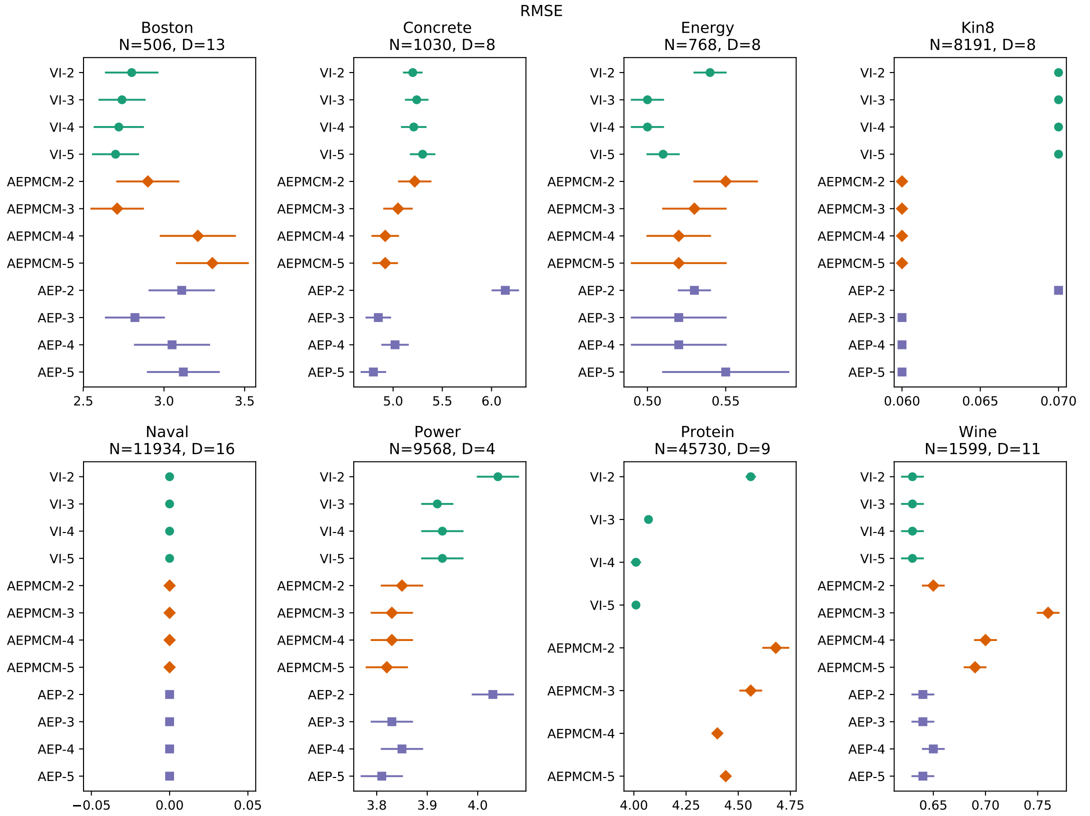


Figure 5.1.1: Test RMSE values plotted with standard errors for all the UCI datasets. Our proposal in brown with a diamond marker for the mean. Lower, to the left is better. Results for the DGP-AEP model using the protein dataset are missing due to excessive computation time.

Experiments results show that all DGP models get similar RMSE scores. When measuring test log-likelihood our DGP model scores the best results in 6 out of the 8 datasets. The models with an intermediate number of layers (3 and 4) perform better, however, no great differences are observed, except for the 2 layers models which perform worse than deeper models.

Table 5.1.1: Regression RMSE (the lower the better) results. Mean for 20 splits and standard errors (in parentheses).

Dataset	N	D	VI-2	VI-3	VI-4	VI-5	AEPMCM-2	AEPMCM-3	AEPMCM-4	AEPMCM-5	AEP-2	AEP-3	AEP-4	AEP-5
Boston	506	13	2.8(0.16)	2.74(0.14)	2.72(0.15)	**2.7(0.14)**	2.9(0.19)	2.71(0.16)	3.21(0.23)	3.3(0.22)	3.11(0.2)	2.82(0.18)	3.05(0.23)	3.12(0.22)
Concrete	1030	8	5.2(0.09)	5.24(0.11)	5.21(0.12)	5.3(0.12)	5.22(0.16)	5.05(0.14)	4.92(0.13)	4.92(0.12)	6.14(0.13)	4.85(0.12)	5.02(0.13)	**4.8(0.12)**
Energy	768	8	0.54(0.01)	**0.5(0.01)**	**0.5(0.01)**	0.51(0.01)	0.55(0.02)	0.53(0.02)	0.52(0.02)	0.52(0.03)	0.53(0.01)	0.52(0.03)	0.52(0.03)	0.55(0.04)
Kin8	8191	8	0.07(0.0)	0.07(0.0)	0.07(0.0)	0.07(0.0)	**0.06(0.0)**	**0.06(0.0)**	**0.06(0.0)**	**0.06(0.0)**	0.07(0.0)	**0.06(0.0)**	**0.06(0.0)**	**0.06(0.0)**
Naval	11934	16	**0.0(0.0)**	**0.0(0.0)**	**0.0(0.0)**	**0.0(0.0)**	**0.0(0.0)**	**0.0(0.0)**	**0.0(0.0)**	**0.0(0.0)**	**0.0(0.0)**	**0.0(0.0)**	**0.0(0.0)**	**0.0(0.0)**
Power	9568	4	4.04(0.04)	3.92(0.03)	3.93(0.04)	3.93(0.04)	3.85(0.04)	3.83(0.04)	3.83(0.04)	3.82(0.04)	4.03(0.04)	3.83(0.04)	3.85(0.04)	**3.81(0.04)**
Protein	45730	9	4.56(0.02)	4.07(0.01)	**4.01(0.02)**	**4.01(0.01)**	4.68(0.06)	4.56(0.05)	4.4(0.02)	4.44(0.01)	-	-	-	-
Wine	1599	11	**0.63(0.01)**	**0.63(0.01)**	**0.63(0.01)**	**0.63(0.01)**	0.65(0.01)	0.76(0.01)	0.7(0.01)	0.69(0.01)	0.64(0.01)	0.64(0.01)	0.65(0.01)	0.64(0.01)

Table 5.1.2: Regression test log-Likelihood (the higher the better) results. Mean for 20 splits and standard errors (in parentheses).

Dataset	N	D	VI-2	VI-3	VI-4	VI-5	AEPMCM-2	AEPMCM-3	AEPMCM-4	AEPMCM-5	AEP-2	AEP-3	AEP-4	AEP-5
Boston	506	13	-2.36(0.04)	-2.3(0.04)	-2.3(0.04)	**-2.28(0.03)**	-2.35(0.06)	-2.32(0.07)	-2.38(0.05)	-2.39(0.07)	-2.41(0.05)	-2.3(0.05)	-2.41(0.07)	-2.43(0.07)
Concrete	1030	8	-3.04(0.02)	-3.05(0.02)	-3.04(0.02)	-3.06(0.02)	-2.93(0.04)	**-2.9(0.04)**	-2.94(0.04)	-2.99(0.05)	-3.1(0.03)	-2.92(0.04)	-2.96(0.03)	-2.93(0.04)
Energy	768	8	-0.82(0.02)	-0.74(0.02)	-0.74(0.02)	-0.76(0.02)	-0.77(0.03)	**-0.68(0.04)**	-0.71(0.04)	**-0.68(0.04)**	-0.9(0.01)	-0.75(0.05)	-0.75(0.05)	-0.72(0.04)
Kin8	8191	8	1.28(0.0)	1.31(0.0)	1.3(0.01)	1.3(0.0)	1.41(0.01)	**1.42(0.01)**	**1.42(0.01)**	1.41(0.01)	1.3(0.01)	1.41(0.01)	1.41(0.01)	**1.42(0.01)**
Naval	11934	16	5.65(0.04)	6.31(0.09)	6.33(0.07)	6.45(0.04)	6.26(0.03)	6.23(0.07)	6.26(0.09)	6.19(0.08)	6.47(0.04)	6.59(0.04)	6.55(0.05)	**6.63(0.05)**
Power	9568	4	-2.82(0.01)	-2.79(0.01)	-2.79(0.01)	-2.79(0.01)	-2.65(0.01)	-2.64(0.01)	**-2.62(0.01)**	-2.63(0.01)	-2.79(0.01)	-2.74(0.01)	-2.74(0.02)	-2.73(0.01)
Protein	45730	9	-2.93(0.0)	-2.82(0.0)	-2.8(0.0)	-2.8(0.0)	-2.26(0.02)	-2.23(0.02)	**-2.15(0.01)**	-2.74(0.01)	-	-	-	-
Wine	1599	11	-0.95(0.01)	-0.95(0.01)	-0.96(0.01)	-0.95(0.01)	-0.98(0.02)	0.44(0.08)	0.95(0.06)	**0.99(0.06)**	-0.98(0.02)	-1.02(0.03)	-1.27(0.07)	-1.73(0.2)

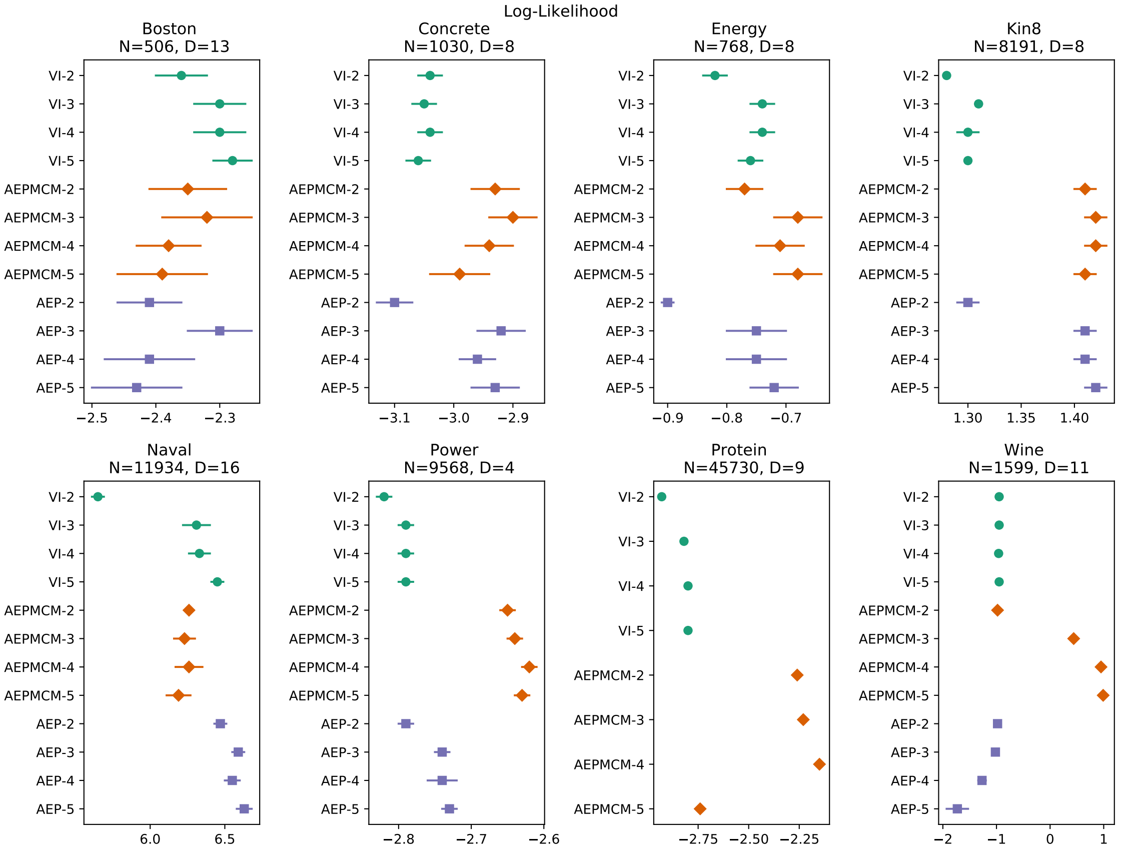


Figure 5.1.2: Test Log-Likelihood values plotted with standard errors for all the UCI datasets. Our proposal in brown with a diamond marker for the mean. Higher, to the right is better. Results for the DGP-AEP model using the protein dataset are missing due to excessive computation time.

5.2 Properties of the predictive distribution

Following [Depeweg et al., 2016] we have generated synthetic data to study the properties of the predictive distribution for the DGP models using Variational inference and Approximate Expectation propagation and Monte carlo methods.

For the first experiment we draw random uniform data for x from the interval $x \in [-2, 2]$ and generate y values from two different functions with probability 0.5:

$$f_1(x) = 10 \sin x + \epsilon \,, \tag{5.2.1}$$

$$f_2(x) = 10 \cos x + \epsilon \,. \tag{5.2.2}$$

Where ϵ is some random Gaussian noise $\epsilon \sim \mathcal{N}(0, 1)$. We train both DGP models with 3 layers, with 3 units in each layer and the same parameters for minibatch size, learning rate and number of inducing points (50, 0.01 and 50 respectively). We also removed the mean function for both methods at the output of the nodes in each layer and train for 500 epochs using Adam [Kingma and Ba, 2014].

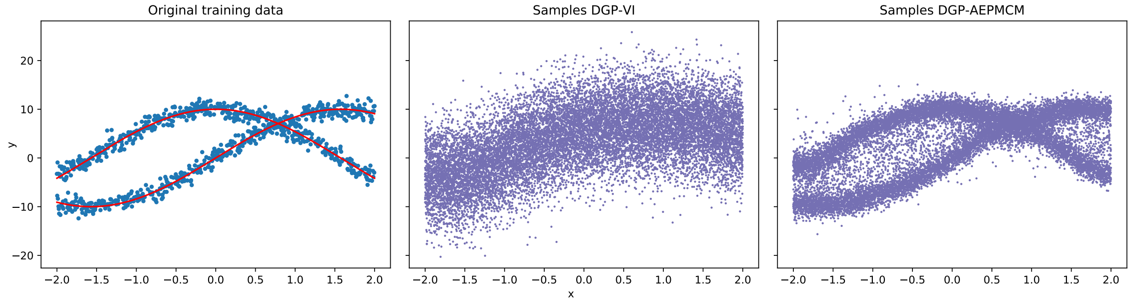


Figure 5.2.1: Comparison of samples from the predictive distribution for the DGP-VI model and DGP-AEPMCM model. The left figure shows the functions (in red) from which the training data (in blue) was generated. The middle figure shows samples of the predictive distribution for the DGP-VI model and the right figure shows samples from the predictive distribution for the DGP-AEPMCM model. Both posterior distributions will yield similar RMSE results, but the DGP-VI model is not able to capture the bimodal nature of the problem.

Figure 5.2.1 shows a comparison of samples taken from the posterior distribution for the DGP-VI model and the DGP-AEPMCM (our proposal) model. Even though both models obtain similar RMSE results, our method is able to successfully capture the bimodal distribution from which the data was generated. This robustness when computing the distribution from which the data is generated can explain the results from Section 5.1, were our model consistently scores the highest values for the test log likelihood. Table 5.2.1 shows RMSE and test log-likelihood results for the sin-cos dataset with a 10% test partition.

Table 5.2.1: Results for the sin-cos synthetic dataset

Model	RMSE	test log-likelihood
DGP-VI	5.293	-3.084
DGP-AEPMCM	5.288	-2.255

To understand why the DGP-AEPMCM obtains higher test log-likelihood scores we can analyze the objective functions for both models. The DGP-VI model is optimizing the Variational Lower bound from Section 4.2.3 which includes the term $\mathbb{E}_q\left[\ln p(y|h^L)\right]$ from (4.2.52). In regression problems a Gaussian likelihood is used. When introduced into the expression above results in something that resembles the RMSE metric. On the other hand, the DGP-AEPMCM model, is optimizing the AEP energy (4.3.5) that has the term $\ln \mathbb{E}_{q^{\setminus L}}\left[p(y|h^{L-1})\right]$. This expression has a similar form to the one of test log-likelihood. With these differences in the objective functions we expect the DGP-AEPMCM model to score better results when measuring test log-likelihood.

For the second experiment the x values are generated uniformly and random from the interval $x \in [-4, 4]$ and the y values are then calculated as:

$$f(x) = 7 \sin x + 3 \, |\cos(x/2)| \, \epsilon \,, \tag{5.2.3}$$

where epsilon is once again some random Gaussian noise.

It is easy to see that the values generated from this function are heteroscedastic, that is, the noise does not remain constant along the x axis.

We trained again two DGP models with the same architecture and parameters as in the first experiment. Figure 5.2.2 shows that the DGP-VI is unable to capture the noise variability and hence the variance of the posterior distribution remains constant along the x axis. The DGP-VI model “averages” the noise of the whole function producing overconfident predictions in noisy regions and under-confident predictions for the relatively noiseless regions. On the other hand, the DGP-AEPMCM model is able to capture the complex noise of the problem varying the variance of the predictions depending on the region of the x space. This is a great advantage when working with real problems as the noise does not need to remain constant. Table 5.2.2 show RMSE and log-likelihood results for the heteroscedastic noise dataset with a 10% test partition.

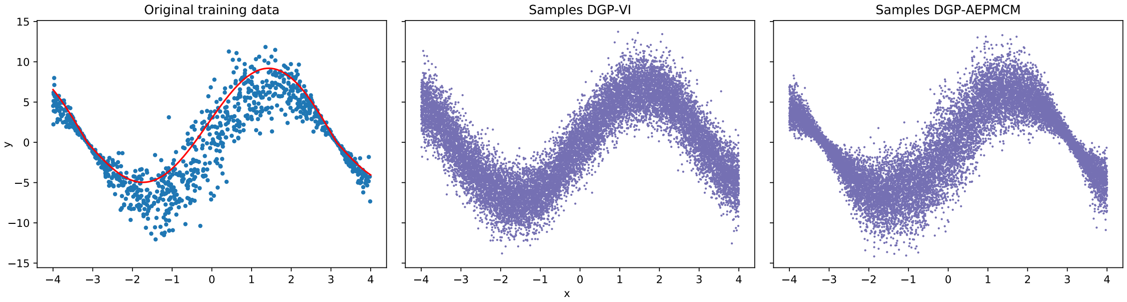


Figure 5.2.2: Comparison of samples from the predictive distribution for the DGP-VI model and DGP-AEPMCM model. The left figure show the function (in red) from which the training data (in blue) was generated. Observe that the noise does not remain constant, creating areas where the noise is significantly reduced. The middle figure shows samples of the predictive distribution for the DGP-VI model and the right figure shows samples from the predictive distribution for the DGP-AEPMCM model.

Table 5.2.2: Results for the heteroscedastic noise synthetic dataset

Model	RMSE	test log-likelihood
DGP-VI	1.905	-2.069
DGP-AEPMCM	2.016	-1.773

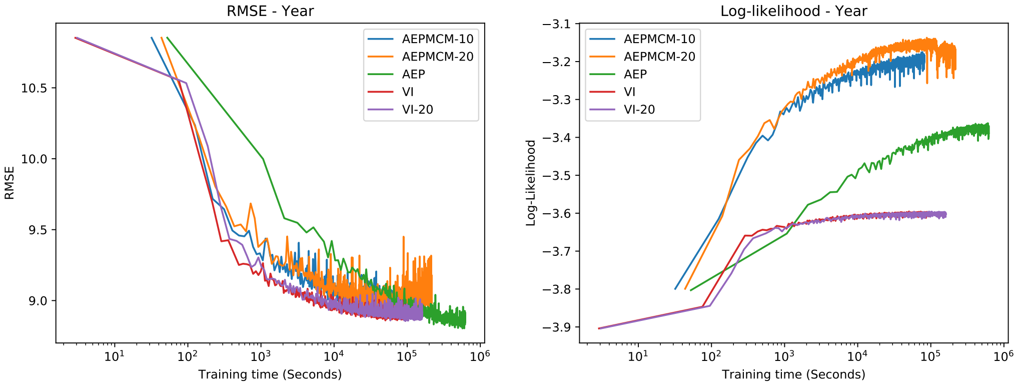
5.3 Big Data experiments

To demonstrate that our proposed method is suitable for Big data problems, we have compared training times of the three state of the art methods for DGPs. For this experiment we have used two Datasets: Year (N = 515,345, D=90) [Bertin-Mahieux et al., 2011] and Airline (N = 2,082,007, D=8). We have trained all models with a minibatch size of 100, using 100 inducing points and all models have a 3 layer architecture with 8 units in each layer.

For models that propagate samples (VI and AEPMCM) we have used two different variants, propagating 10 and 20 samples for training. We trained using Adam optimizer with a learning rate of 0.01 and we have measured training times every 100 gradient steps.

For this experiment we use the same CPU (using all 6 cores) for all models: “Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz”.

For the Year dataset we have followed instructions from the authors Bertin-Mahieux et al. [2011] and used the first 463,715 examples for train and the remaining for test. Figures 5.3.1a and 5.3.1b shows graphs of the RMSE and Log-likelihood respectively, against training time. A summary of the results are shown in Table 5.3.1. Final RMSE and test log-likelihood results have been calculated as the median values of the last 2000 iterations (due to the noise observed in the values).



(a) (b)

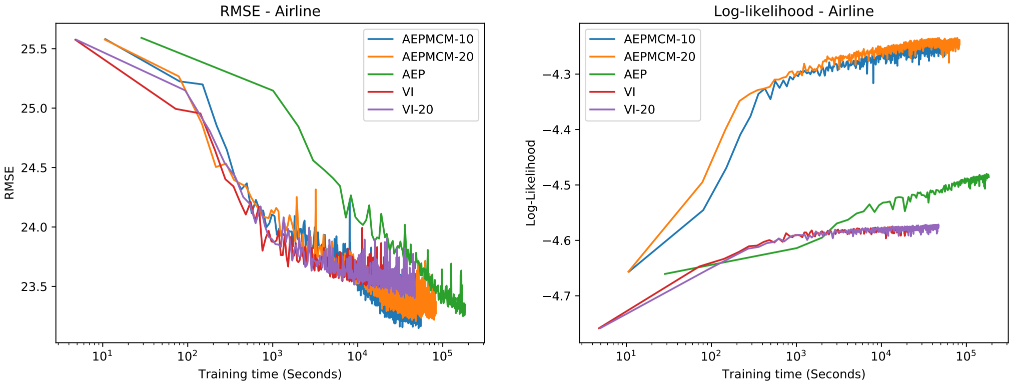
Figure 5.3.1: Results for RMSE (Left, the lower the better) and Log-likelihood (right, the higher the better) for the dataset Year, against training time (in seconds, log scale) measured every 100 gradient steps.

Table 5.3.1: Results for the Big data experiments. Year

Model	avg. gradient step (seconds)	Final RMSE	Final Log-Likelihood
DGP-AEPMCM-10	0.0281	8.95	-3.19
DGP-AEPMCM-20	0.0458	9.04	-3.17
DGP-VI-10	0.0211	8.89	-3.60
DGP-VI-20	0.0403	8.90	-3.60
DGP-AEP	0.2940	8.85	-3.37

For the Airline dataset we have chosen a 10% random partition for test. Results are shown in Table 5.3.2 and Figures 5.3.2a and 5.3.2b.

Results show that all models obtain a similar RMSE score. The average time to compute a single gradient step is similar for both models than propagate samples, with the doubly stochastic VI approach taking slightly less time to compute a gradient step. We see that when doubling the number of samples to use, both models roughly double the computing times. The code for our model allows the inducing points to be different for each unit in the hidden layers, increasing the flexibility at the cost of higher computing times. However, this functionality was disabled through all the experiments. The higher time for computing a single gradient step in the DGP-AEP model is due to the need of computing the expected value with respect to kernel functions from (4.2.24) and (4.2.25),



(a) (b)

Figure 5.3.2: Results for RMSE (Left figure, the lower the better) and Log-likelihood (right figure, the higher the better) for the Airline dataset, against training time (in seconds, log scale) measured every 100 gradient steps.

Table 5.3.2: Results for the Big data experiments. Airline

Model	avg. gradient step (seconds)	Final RMSE	Final Log-Likelihood
DGP-AEPMCM-10	0.0221	23.22	-4.25
DGP-AEPMCM-20	0.0347	23.32	-4.24
DGP-VI-10	0.0202	23.55	-4.58
DGP-VI-20	0.0388	23.47	-4.57
DGP-AEP	0.2914	23.32	-4.48

which are complex expressions that require more time to be computed.

However, when comparing log-likelihood values, our approach performs significantly better in both experiments. A notable difference is that the sampling procedure for the expectation in the Doubly stochastic approach is unbiased [Salimbeni and Deisenroth, 2017], while the sampling procedure in our proposal is biased. This is due to the non-linearity of the logarithm. As the number of samples grows, results are expected to improve for the AEPMCM model, this can be seen in the plots, Figures 5.3.1b and 5.3.2b.

Chapter 6

Conclusions and Future Work

In this master thesis we have shown that GP models are attractive alternatives to classic machine learning models, offering uncertainty estimates over the predictions and nonparametric modeling. We have also seen that the high training complexity that scales cubically with the number of training examples can be overcome by using sparse approximations which reduce the complexity without significantly reducing the accuracy of the predictions.

It has been demostrated that DGP models improve the results of shallow GPs while maintaining the properties that make them attractive in the first place. However, DGP models are hard to train and direct Bayesian inference is not usually possible. In recent years there has been some proposals that use approximate inference techniques to train DGP models. We have covered two of the most used techniques: Variational inference and Expectation propagation, explaining the advantages and drawbacks of each of them.

The method proposed in this thesis extends the work done in [Bui et al., 2016] by removing the assumption that the output distribution needs to be of a fixed form. Our approach uses Monte Carlo Methods to sample from the output at each layer and propagates those samples through the network. This allows to better capture the properties of the outputs distributions.

We have shown through our experiments that our approach improves the results over the current state of the art inference technique for DGPs using Expectation Propagation and that is capable of modeling complex data such as multi-modal data unlike the Variational inference approach. At the same time we have proved that our method is suitable for large datasets due to the sparse GPs representation used and the stochastic gradient descent techniques (such as minibatch training), reducing memory usage and training time.

As an improvement to our method and one of the possible areas for future work is to use the same approach for binary classification and multi-class problems. Using Bayesian methods (as the one proposed) for classification provides not only a value for the class prediction but also a uncertainty estimate (as in the regression case) for the prediction. Unfortunately this involves changing the Gaussian likelihood of the model and using a different one which will require modifications at the output of our DGP network.

It would also be possible to adapt the proposed model for multi-task learning. This involves problems in which multiple tasks that share some common structure need to be solved. By including information about the different task one can expect better results than tackling them individually. Due to the flexibility of using a deep learning approach it is possible to solve these tasks with a DGP network architecture in which there are some common layers for all the tasks (typically the inner layers would be shared) jointly with specific output layers (which may be different) for each of the different tasks.

A different line of future work that could improve the results of the proposed method is removing the hypothesis that the approximate posterior distributions are Gaussian and use more complex distributions instead. There are different approaches in the literature that could be used for this:

- The work in [Liu, 2017] proposes to use a set of particles for which locations can be optimized using a functional gradient descent technique. This is effectively minimizing the KL divergence between the particles and the true posterior. After this phase the particles are expected to fit true posterior distribution form. This technique allows the particles to fit any distribution without imposing a restriction in the form of the approximation.

- The authors in [Rezende and Mohamed, 2015] propose to use Normalizing flows to approximate the posterior distribution which allows to construct more complex distributions. By transforming a probability distribution by applying a series of invertible mappings (compositions of functions) it is possible to transform a simple probability distribution, like for example a Gaussian, into a more complex one with (possibly) multiple modes.

- The approach in [Mescheder et al., 2017] shows that it is possible to use a black-box approximation whose parameters are trained using adversarial training (as in Generative Adversarial Networks) to create more expressive approximations. They apply this technique to Variational autoencoders, but a similar approach could be taken to approximate distributions for other models.

- Authors in [Li et al., 2017] propose a different approach to improve the approximation q for a true distribution p. The main idea is that, to train a sampler that best approximates the true distribution p, a generator can generate samples that are then presented to a "teacher". The "teacher" checks whether the samples are approximating the true posterior and then improves those samples by running MCMC. Finally, the generator takes the improved samples and use them to enhance the generation process (using an update rule for its parameters) for the next iteration. This process is run until the "teacher" is no longer able to improve the samples, which means that the generator has converged to the stationary function of the MCMC algorithm, which is the true posterior.

Appendices

Appendix A

Gaussian identities

The following identities are taken from [Roweis, 1999]

The univariate normal distribution can be written as

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{1.0.1}$$

Extending the univariate to D-dimensional variables we have the multivariate normal distribution

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\mathbf{\Sigma}) = (2\pi)^{-D/2}|\mathbf{\Sigma}|^{1/2} \exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})) \tag{1.0.2}$$

Where $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ are the parameters of the distribution, mean and covariance respectively. If $\mathbf{x}$ and $\mathbf{y}$ have jointly a Gaussian distribution

$$\mathbf{z} = \begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix}, \begin{bmatrix}\mathbf{A} & \mathbf{C}\\ \mathbf{C}^T & \mathbf{B}\end{bmatrix}\right) \tag{1.0.3}$$

the marginal distributions $p(\mathbf{x})$, $p(\mathbf{y})$ are given by

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{a},\mathbf{A}) \tag{1.0.4}$$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{b},\mathbf{B}) \tag{1.0.5}$$

The conditional distributions are

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{a}+\mathbf{C}\mathbf{B}^{-1}(\mathbf{y}-\mathbf{b}), \mathbf{A}-\mathbf{C}\mathbf{B}^{-1}\mathbf{C}^T) \tag{1.0.6}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{b}+\mathbf{C}^T\mathbf{A}^{-1}(\mathbf{x}-\mathbf{a}), \mathbf{B}-\mathbf{C}^T\mathbf{A}^{-1}\mathbf{C}) \tag{1.0.7}$$

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Agathe Girard, Carl Edward Rasmussen, C. J. Q. n. and Murray-Smith, R. (2003). Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 545–552. MIT Press.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. (2011). The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, Secaucus, NJ, USA.

Bui, T. D., Hernández-Lobato, J. M., Hernández-Lobato, D., Li, Y., and Turner, R. E. (2016). Deep gaussian processes for regression using approximate expectation propagation. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, pages 1472–1481.

Cornford, D., Nabney, I. T., and Williams, C. K. I. (1998). Adding constrained discontinuities to gaussian process models of wind fields. In *Proceedings of the 11th International Conference on Neural Information Processing Systems*, pages 861–867, Cambridge, MA, USA. MIT Press.

Cutajar, K., Bonilla, E. V., Michiardi, P., and Filippone, M. (2017). Random feature expansions for deep Gaussian processes. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 884–893. PMLR.

Damianou, A. (2015). Deep gaussian processes and variational propagation of uncertainty. *PhD Thesis, University of Sheffield.*

Damianou, A. and Lawrence, N. (2013). Deep gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 207–215. PMLR.

Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., and Udluft, S. (2016). Learning and policy search in stochastic dynamical systems with bayesian neural networks. *CoRR*, abs/1605.07127.

Duvenaud, D. (2014). *Automatic model construction with Gaussian processes.* PhD thesis, University of Cambridge.

Duvenaud, D., Lloyd, J., Grosse, R., Tenenbaum, J., and Zoubin, G. (2013). Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1166–1174. PMLR.

Duvenaud, D., Rippel, O., Adams, R., and Ghahramani, Z. (2014). Avoiding pathologies in very deep networks. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 202–210. PMLR.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian data analysis.* Chapman and Hall/CRC.

Higham, N. J. (1996). *Accuracy and Stability of Numerical Algorithms.* Society for Industrial and Applied Mathematics.

Hinton, G. E. (2009). Deep belief networks. *Scholarpedia.*

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR).*

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc.

Klaus Greff, Aaron Klein, Martin Chovanec, Frank Hutter, and Jürgen Schmidhuber (2017). The Sacred Infrastructure for Computational Research. In Katy Huff, David Lippa, Dillon Niederhut, and Pacer, M., editors, *Proceedings of the 16th Python in Science Conference*, volume 28 of *NIPS'15*, pages 49 – 56.

Ko, J. and Fox, D. (2009). Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2017). Deep neural networks as gaussian processes.

Li, Y., Hernandez-Lobato, J. M., and Turner, R. E. (2015). Stochastic expectation propagation. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2323–2331.

Li, Y., Turner, R. E., and Liu, Q. (2017). Approximate inference with amortised MCMC. *CoRR*, abs/1702.08343.

Liu, Q. (2017). Stein variational gradient descent as gradient flow. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 3115–3123. Curran Associates, Inc.

MacKay, D. J. C. (1996). Bayesian methods for backpropagation networks. In *Models of Neural Networks III: Association, Generalization, and Representation*, pages 211–254. Springer.

Mescheder, L., Nowozin, S., and Geiger, A. (2017). Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *International Conference on Machine Learning (ICML)*.

Minka, T. P. (2001a). The ep energy function and minimization schemes.

Minka, T. P. (2001b). Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pages 362–369, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Neal, R. M. (1996a). *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg.

Neal, R. M. (1996b). Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pope, C. A., Gosling, J. P., Barber, S., Johnson, J., Yamaguchi, T., Feingold, G., and Blackwell, P. (2018). Modelling spatial heterogeneity and discontinuities using voronoi tessellations. *ArXiv e-prints*.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. The MIT Press.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France. PMLR.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China. PMLR.

Roweis, S. (1999). Gaussian identities. *Lectures Notes, University of Toronto.¡ http://www. cs. toronto. edu/roweis/notes/gaussid. pdf*.

Salimbeni, H. and Deisenroth, M. (2017). Doubly stochastic variational inference for deep gaussian processes. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4588–4599. Curran Associates, Inc.

Seeger, M. (2005). Expectation propagation for exponential families. Technical report.

Seeger, M., Williams, C., and Lawrence, N. (2003). Fast forward selection to speed up sparse gaussian process regression. In *Artificial Intelligence and Statistics 9*, number EPFL-CONF-161318.

Smola, A. J. and Bartlett, P. L. (2001). Sparse greedy gaussian process regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press.

Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems 18*, pages 1257–1264. MIT Press.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Vafa, K. (2016). *Training and Inference for Deep Gaussian Processes*. PhD thesis, Harvard College.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.

Weiss, N., Holmes, P., and Hardy, M. (2006). *A Course in Probability*. Pearson Addison Wesley.