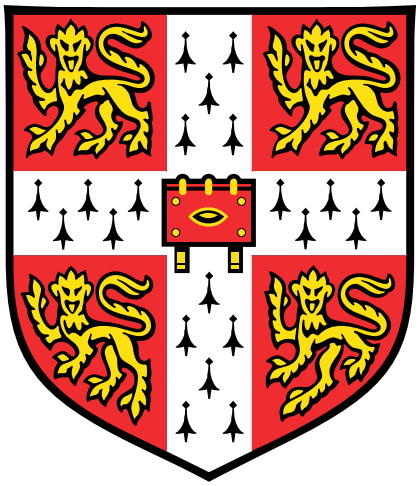
Pathologies of Deep Sparse Gaussian Process Regression



Sergio Pascual Díaz

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Fitzwilliam College

August 2017

Declaration

I, Sergio Pascual Díaz of Fitzwilliam College, being a candidate for the M.Phil in Machine Learning, Speech and Language Technology, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. This dissertation contains fewer than 8,000 words. Signed:

Sergio Pascual Díaz
August 2017

Acknowledgements

First, I wish to thank my supervisor, Dr. Richard Turner, for his indispensable guidance and support throughout the project. Secondly, I would like to thank Thang Bui for being a helping hand whenever needed. Last but not least, I want to thank my parents, Jose Angel and Ana Cruz, as well as my sister, Nerea, for cheering me up and offering their unconditional support and love.

Abstract

Deep Gaussian Processes (DGPs) are a powerful multi-layer hierarchical generalisation of Gaussian Processes. However, the composition of non-linear layer mappings makes learning and inference analytically intractable, thus requiring approximate inference. Recently, sparse DGPs demonstrated state-of-the-art performance on several large-scale datasets for the first time using an inference scheme that combines FITC sparse GP approximations and an approximate Expected Propagation scheme. With the goal of investigating a potential implementation of sparse DGPs in Bayesian Optimisation, we study the performance of this method in some challenging medium-size synthetic datasets. In addition to reporting the pathological behaviour encountered, we propose a greedy initialisation scheme that solves it and brings the use of DGPs in Bayesian Optimisation one step closer.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

In this project, we investigate the pathologies encountered in sparse Deep Gaussian Process regression for medium-size datasets. We start by discussing the limitations of Gaussian Processes, a popular Bayesian non-parametric model used in many areas of supervised learning.

1.1 Limitations of Gaussian Processes

Part of the popularity of *Gaussian Processes* (GPs) relies on a simple and elegant inference and learning framework that provides well-calibrated uncertainty estimates. However, as non-parametric models the expressiveness of GPs is constrained by the choice of covariance function, which encapsulates our prior modelling assumptions. In complex datasets, where the underlying function presents non-stationary behaviour, heteroscedastic (input-dependent) noise or dependencies between output dimensions, standard GPs are known to have limited predictive performance.

To address these limitations, many variants of GPs have been proposed in recent literature. Some approaches require sophisticated hand-designed covariance functions, Durrande et al. (2012), which only guarantee local smoothing at fixed length-scales. In addition, hyperparameter tuning becomes non-trivial in those cases. Other approaches include: input and output wrapping, Shahriari et al. (2015); and convolution processes for leverage of multiple correlated outputs, Alvarez et al. (2011).

1.2 History of Deep Gaussian Processes

A well-known fact about Gaussian Processes is that a neural network with an infinitely wide hidden-layer and unknown weights is formally equivalent to a GP as shown in Warner and Neal (1997). Motivated by this fact and the recent success of deep architectures in a variety of challenging problems in probabilistic modelling, Damianou and Lawrence (2013) introduced *Deep*

Gaussian Processes (DGPs) as a multi-layer hierarchical generalisation of Gaussian Processes.

While preserving the non-parametric nature of GPs, the hierarchical structure of DGPs allows for multiple-level abstract representations of complex datasets characteristic of deep models. Hence by performing automatic input wrapping, data compression/expansion and non-parametric kernel design, DGPs can potentially overcome the limitations of standard GPs mentioned above.

Approximate inference is required for DGPs as the composition of non-linearities through the GP-layers makes inference and learning analytically intractable. Damianou and Lawrence (2013) presented a variational-free-energy approach inspired by the variational sparse approximation of GPs of Titsias (2009). However, this method presents problems in small datasets, where initialisation is known to be an issue, as well as, large datasets since number of variational parameters to optimise in this model increases linearly with the number of training points. Later, Hensman and Lawrence (2014) proposed a variational nested scheme that solved the latter issue.

More recently, Bui et al. (2016) presented an inference scheme for sparse DGPs based on the combination of the FITC (Full Independent Training Conditional) sparse approximation of GPs with an approximate Expected Propagation (AEP) inference. This approach has proven to be a flexible, scalable and well-calibrated regression model, demonstrating state-of-the-art performance in several of large-size datasets.

In addition, Duvenaud et al. (2014) reported a pathology in the representational capacity of DGPs with a single hidden-unit per layer. As the number of layers increases, the composition of GP mappings concentrates the predicted density onto one-dimensional curves, making DGPs unsuitable for modelling higher-dimensional manifolds. This pathology can be fixed by increasing the number of hidden units per layer or by introducing an input-connected architecture as Duvenaud et al. proposed.

1.3 Challenging problems in Bayesian Optimisation

Bayesian Optimisation aims to find global optima of black-box functions, where evaluation is expensive, noisy and generally gradient information is not available. Black-box functions are unknown, non-convex functions that often exhibit complex non-stationary behaviour. Based on current data available, Bayesian Optimisation employs probabilistic models to predict the shape of the objective function and guide an efficient data collection strategy.

Bayesian non-parametric models, such as Gaussian Processes, are a popular choice for such tasks as they can incorporate prior beliefs about the objective and iteratively update them based on new queries. GPs also provide uncertainty estimates over the predicted objective

function, something specially useful in reduced data domains typically encountered in Bayesian Optimisation. In addition, the analytic properties of the predictive distribution of GPs make certain choices of acquisition function, such as Expected Improvement (EI), tractable.

Part of the motivation behind this thesis was to investigate the implementation of Deep Gaussian Processes for Bayesian Optimisation. Apart from potentially overcoming many of the modelling limitations of GPs, DGPs could leverage correlations between multiple outputs and exploit evaluations on cheap surfaces to estimate expensive correlated objectives. In this thesis, we make a first step and study the performance of sparse DGPs on one-dimensional medium-size synthetic datasets that exhibit some of the properties expected in challenging higher-dimensional problems in Bayesian Optimisation.

1.4 Thesis contribution

Deep Gaussian Processes (DGPs) are flexible Bayesian non-parametric models with a high representational power. In this thesis, we study the predictive performance of the approximate Expected Propagation (AEP) inference scheme for sparse DGPs proposed by Bui et al. (2016), which has demonstrated state-of-the-art results on several large-scale datasets.

We start by reviewing the full AEP-DGP model, emphasizing the characteristics and limitations of the approximations it is based upon (Chapter 2). Then, we study the modelling capacity of AEP-DGPs on benchmark toy datasets, such as the step function, where full GPs are known to struggle (Chapter 3).

Later, we test the predictive performance of AEP-DGPs on medium-size datasets generated from DGP samples that exhibit variable length-scales. We investigate the pathologies encountered in AEP-DGP regression, when compared with sparse and full GPs, by revising the training procedure and proposing alternative configurations (Chapter 4).

In addition, we introduce a sparse GP greedy initialisation scheme for the DGP layers that resolves this pathological behaviour by facilitating the propagation from input to output. We compare this and other initialisation schemes by evaluating their performance on out-of-sample test sets (Chapter 5).

For our experiments, we make use of the Python library `geepee`[1], which includes a full implementation of the AEP-DGP model.

[1] https://github.com/thangbui/geepee

Chapter 2

Theoretical Framework

In this chapter, we set the theoretical framework required for the discussion of experiment results in future chapters. First, we give a formal description of Gaussian Processes (GPs) and extend it to the probabilistic model of a full Deep Gaussian Process (DGP). Then, we explain in detail the approximate inference scheme for sparse DGP regression proposed in Bui et al. (2016) and implemented the `geepee` library. Finally, we describe the evaluation metrics used in future experiments for model comparison.

2.1 Generalising Gaussian Processes

A *Gaussian Process* is non-parametric generative model that yields distributions over functions. Formally, a Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution, Rasmussen and Williams (2004). In other words, the joint distribution of GP function evaluations $f(\mathbf{x}_1), \cdots, f(\mathbf{x}_N) \in \mathbb{R}$ at inputs $\mathbf{x}_i \in \mathbb{R}^{D^{in}}$ for $i = 1, \cdots, N$ is given by the multivariate Gaussian $f_{1:N} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}_{ff})$. Here, we compute the mean vector $\mathbf{m} = (m_\theta(\mathbf{x}_1), \cdots, m_\theta(\mathbf{x}_N)) \in \mathbb{R}^N$ from a mean function $m_\theta(\mathbf{x}) = \mathbb{E}(f(\mathbf{x}))$, and the $N \times N$ covariance matrix with entries $(\mathbf{K}_{ff})_{ij} = \mathcal{K}_\theta(\mathbf{x}_i, \mathbf{x}_j)$ from a symmetric positive-definite covariance (or kernel) function $\mathcal{K}_\theta(\mathbf{x}, \mathbf{x}') = Cov(f(\mathbf{x}), f(\mathbf{x}'))$. For simplicity, we denote a Gaussian Process prior over a function f as $f \sim \mathcal{GP}(m(\cdot), \mathcal{K}(\cdot, \cdot))$.

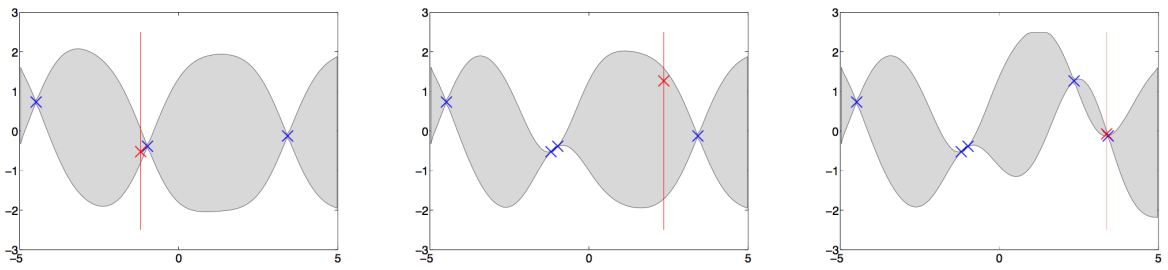


Figure 2.1 Three updates of the Gaussian Process posterior distribution.

2.1.1 Gaussian Process regression

In Gaussian Process regression given inputs $\mathbf{x}_i \in \mathbb{R}^{D_{in}}$, observations are modelled as $y_i = f(\mathbf{x}_i) + \epsilon_i$ for $i = 1, \cdots, N$, where $f \sim \mathcal{GP}(0, \mathcal{K}(\cdot,\cdot))$ and $\epsilon_i \sim \mathcal{N}(0, \sigma^2_{out})$ is the homoscedastic (input independent) Gaussian noise term. Thus, the data likelihood $p(y_{1:N}|\mathbf{x}_{1:N}, f) = \mathcal{N}(y_{1:N}; f_{1:N}, \sigma^2_{out} I)$. By conditioning on the dataset, we can compute the posterior over f and analytically derive a closed-form expression for the predictive distribution at a new test point $\mathbf{x}^*$:

$$p(y^*|\mathbf{x}^*, y_{1:N}, \mathbf{x}_{1:N}, \alpha) = \mathcal{N}(y^*|\mu_{pred}(\mathbf{x}^*), \sigma^2_{pred}(\mathbf{x}^*)) \tag{2.1}$$

where $\alpha = (\theta,\, \sigma^2_{out})$ is the set of hyperparameters and:

$$\begin{aligned} \mu_{pred} &= \mathbf{k}^T_{f^*f}(\mathbf{K}_{ff} + \sigma^2_{out} I)^{-1} y_{1:N} \\ \sigma^2_{pred} &= \mathcal{K}(\mathbf{x}^*, \mathbf{x}^*) + \sigma^2_{out} - \mathbf{k}^T_{f^*f}(\mathbf{K}_{ff} + \sigma^2_{out} I)^{-1}\mathbf{k}_{f^*f} \\ \text{with } (\mathbf{k}_{f^*f})_i &= \mathcal{K}(\mathbf{x}*, \mathbf{x}_i) \text{ for } i = 1, \cdots, N \end{aligned} \tag{2.2}$$

Similarly, the objective function used for model comparison and hyperparameter tuning, the marginal likelihood, is also Gaussian:

$$p(y_{1:N}|\mathbf{x}_{1:N}, \alpha) = \mathcal{N}(y_{1:N}; \mathbf{K}_{ff} + \sigma^2_{out} I) \tag{2.3}$$

The inversion of the covariance matrix in (2.2) elevates the computational cost of a full GP model to $\mathcal{O}(N^3)$ making inference slow and even intractable for large-scale datasets.

2.1.2 Deep Gaussian Process model

Deep Gaussian Processes were introduced in Damianou and Lawrence (2013) as a non-parametric deep generative model that generalises Gaussian Processes. Damianou et al. proposed a multi-layer hierarchical structure where the mappings between layers are modelled by Gaussian Processes.

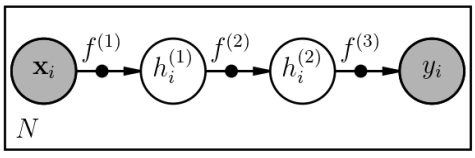


Figure 2.2 Graphical model of a full DGP with 2 hidden-layers connected by GP mappings $f^{(l)}$.

Given data inputs $\mathbf{x}_i \in \mathbb{R}^{D_{in}}$ and observations $y_i \in \mathbb{R}^{D_{out}}$ for $i = 1, \cdots, N$, we denote the l-th hidden layer of size D_l with layer noise $\sigma^2_{(l)}$ as the N-vector $h^{(l)}_{1:N} = (h^{(l)}_1, \cdots, h^{(l)}_N) \in \mathbb{R}^{N \times D_l}$ for $l = 2, \cdots, L-1$ [1]. Then, the generative process a full L-layer DGP is a composition of

[1] Sometimes we may refer to $\mathbf{x}_i$ as $h^{(1)}_i$, y_i as $h^{(L)}_i$ and σ^2_{out} as $\sigma^2_{(L)}$.

Gaussian Process mappings $f^{(l)} \sim \mathcal{GP}(\mathbf{0}, \mathcal{K}^{(l)}(\cdot,\cdot))$ for $l = 1, \cdots, L$:

$$p(h^{(1)}_{1:N}|f^{(1)}, \mathbf{x}_{1:N}, \sigma^2_{(1)}) = \prod_i^N \mathcal{N}(h^{(1)}_i|f^{(1)}(\mathbf{x}_i), \sigma^2_{(1)}) \text{ – input layer} \tag{2.4}$$

$$p(h^{(l)}_{1:N}|f^{(l)}, h^{(l-1)}_{1:N}, \sigma^2_{(l)}) = \prod_i^N \mathcal{N}(h^{(l)}_i|f^{(l)}(h^{(l-1)}_i), \sigma^2_{(l)}) \text{ – for } l = 2, \cdots, L-1$$

$$p(y_{1:N}|f^{(L)}, h^{(L-1)}_{1:N}, \sigma^2_{out}) = \prod_i^N \mathcal{N}(y_i|f^{(L)}(h^{(L-1)}_i), \sigma^2_{out}) \text{ – output layer}$$

In this project, the covariance functions of GP mappings, $\mathcal{K}^{(l)}(\cdot,\cdot) : \mathbb{R}^{D_{l-1}} \times \mathbb{R}^{D_{l-1}} \mapsto \mathbb{R}$, are set to be from the standard squared exponential (SE) kernel family with:

$$\mathcal{K}^{(l)}(\mathbf{x}, \mathbf{x}') = v^2_{(l)} \exp(-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2l^2_{(l)}})$$

For $L > 1$, the composition of non-linear mappings between layers makes inference of the posterior analytically intractable. Hence, approximate inference is required in order to approximate the posterior over GP mappings, $f^{(l)}$, and produce an estimate of the marginal likelihood $p(y_{1:N}|\mathbf{x}_{1:N}, \alpha)$ used in hyperparameter $\alpha = \{l_{(l)}, v_{(l)}, \sigma^2_{(l)}\}_{l=1}^L$ learning.

2.2 Sparse Deep Gaussian Processes

In this project, we test and investigate the performance of the approximate inference scheme for Deep Gaussian Process regression presented in Bui et al. (2016). This technique has shown state-of-the-art performance in several large-scale datasets, while providing a fast, flexible and well-calibrated approximate DGP inference framework.

2.2.1 FITC sparsification

For this inference scheme, GP-mappings between the layers of a DGP, $f^{(l)}$-s, are approximated using FITC (Fully Independent Training Conditional), Quiñonero-candela et al. (2005) and Snelson and Ghahramani (2006), a pseudo-point based sparse approximation of GPs. Consider the l-th DGP layer $h^{(l)}_i = f^{(l)}(h^{(l-1)}_i) + \epsilon_i \in \mathbb{R}^{D_l}$, with $\epsilon_i \sim \mathcal{N}(0, \sigma^2_{(l)})$ for $i = 1, \cdots, N$. FITC

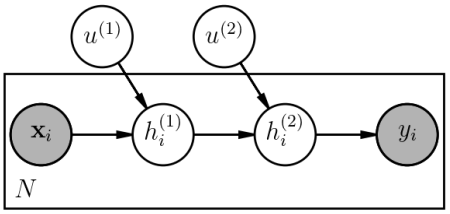


Figure 2.3 Graphical model of a two-hidden-layer FITC-DGP with inducing outputs $u^{(l)}$.

creates a semi-parametric model by inducing a small set of $M << N$ pseudo-outputs $\{u^{(l)}_j\}_{j=1}^M$ in $\mathbb{R}^{D_l}$, the infinite output space of $f^{(l)}$. By assuming that other function evaluations, $f^{(l)}(\cdot)$, are conditionally independent from each other given $u^{(l)}_{1:M}$, we can construct an approximate

probabilistic model for the DGP hidden-layers $l = 1, \cdots, L$:

$$p(u^{(l)}_{1:M}) = \mathcal{N}(u^{(l)}_{1:M}; \mathbf{0}, \mathbf{K}^{(l)}_{uu}) \tag{2.5}$$

$$p(h^{(l)}_{1:N} | u^{(l)}_{1:M}, h^{(l-1)}_{1:N}, \sigma^2_{(l)}) = \prod_i^N \mathcal{N}(h^{(l)}_i | \mathbf{C}^{(l)}_i u^{(l)}_{1:M}, \Sigma^{(l)}_i) \tag{2.6}$$

where $(\mathbf{K}^{(l)}_{uu})_{ij} = \mathcal{K}^{(l)}(z^{(l)}_i, z^{(l)}_j)$ evaluated at layer pseudo-inputs $z^{(l)}_j \in \mathbb{R}^{D_{l-1}}$ whose location can be chosen by optimizing an estimate of the marginal likelihood. In addition, we define matrices:

$$\mathbf{C}^{(l)}_i = \mathbf{K}^{(l)}_{h_i u} \mathbf{K}^{(l)-1}_{uu} \tag{2.7}$$

$$\Sigma^{(l)}_i = \mathbf{K}^{(l)}_{h_i h_i} + \sigma^2_{(l)} I - \mathbf{C}^{(l)}_i \mathbf{K}^{(l)}_{u h_i} \tag{2.8}$$

$$\text{with } (\mathbf{K}^{(l)}_{h_i u})_{nm} = \mathcal{K}^{(l)}(h^{(l-1)}_{i,m}, z^{(l)}_n) \text{ and } (\mathbf{K}^{(l)}_{h_i h_i})_{nm} = \mathcal{K}^{(l)}(h^{(l-1)}_{i,m}, h^{(l-1)}_{i,n}) \tag{2.9}$$

The computational cost of inference and learning associated with the FITC method is $\mathcal{O}(NM^2)$, tractable as long as M remains relatively small compared to N. However, the FITC approximation induces a heteroscedastic (non-stationary) noise on the layer outputs, capturing the uncertainty produced by the sparsification, which propagates through the DGP network. In future chapters, we will discuss some of the subtleties when training FITC sparse GP layers, including number of induced pseudo-points per layer, the impact of adding more pseudo-points and the initial location the pseudo-inputs.

The composition of non-linearities through the DGP layers makes the posterior over inducing outputs $p(\mathbf{u}|\mathbf{x}_{1:N}, y_{1:N})$ analytically intractable, requiring approximate inference. Next, we present a inference scheme that approximates the posterior and yields an approximation to the estimate marginal likelihood $p(y_{1:N}|\mathbf{x}_{1:N}, z^{(1:L)}, \alpha)$.

2.2.2 Approximate Expected Propagation (AEP)

Approximating the posterior $p(\mathbf{u}|\mathbf{x}_{1:N}, y_{1:N}) \propto p(\mathbf{u}) \prod_{i=1}^N p(y_i|\mathbf{x}_i, \mathbf{u})$ with $q(\mathbf{u}) = p(\mathbf{u}) \prod_{i=1}^N t_i(\mathbf{u})$ using Expectation Propagation (EP), Minka (2001), yields an EP-energy function that can be optimised for DGP training. However, storing natural parameters θ_i of Gaussian factors $t_i(\mathbf{u})$ has a high computational cost of $\mathcal{O}(NLM^2)$ for large datasets.

Alternatively, Stochastic EP (SEP) in Li et al. (2015) adds a tied factor constraint to EP by setting $t_i(\mathbf{u}) = t(\mathbf{u})$ for all $i \in \{1, \cdots, N\}$, which reduces the cost in memory to $\mathcal{O}(LM^2)$. The tied factor $t(\mathbf{u})$ captures the average effect of the likelihood. The negative of the resulting SEP-energy function was proposed byBui et al. (2016) as objective function for DGP training:

$$\mathcal{F} = (1 - N)\phi(\theta) - \phi(\theta_{prior}) + N\phi(\theta^{\backslash 1}) + \sum_{i=1}^N \log \mathcal{Z}_i \tag{2.10}$$

$$\text{where } \mathcal{Z}_i = \int_{\mathbf{u}} p(y_i|\mathbf{x}_i, \mathbf{u}) q^{\backslash 1}(\mathbf{u}) d\mathbf{u} = \int_{\mathbf{u}} p(y_i|\mathbf{x}_i, \mathbf{u}) p(\mathbf{u}) t^{N-1}(\mathbf{u}) d\mathbf{u} \tag{2.11}$$

Here θ_{prior}, θ^1, $\theta = \theta_{prior} + N\theta^1$ and $\theta^{\backslash 1} = \theta - \theta^1$ are the natural parameters of unnormalised Gaussians $p(\mathbf{u}), t(\mathbf{u}), q(\mathbf{u})$ and the cavity $q^{\backslash 1}(\mathbf{u}) = q(\mathbf{u})/t(\mathbf{u})$ (respectively) and $\phi(\lambda)$ is the log

normaliser of a Gaussian with natural parameters λ.

Interestingly, the approximation of marginal log-likelihood in (2.10) is a particular case of the Black-Box-α energy function when $\alpha = 1$, see Hernández-Lobato et al. (2015):

$$E_{BB-\alpha}(\theta, \theta_{prior}) = \phi(\theta_{prior}) - \phi(\theta) - \frac{1}{\alpha}\sum_{i=1}^{N} \log \mathbb{E}_{q(\mathbf{u})}[(p(y_i|\mathbf{x}_i, \mathbf{u})/t(\mathbf{u}))^{\alpha}] \tag{2.12}$$

and $E_{BB-1}(\theta, \theta_{prior}) = -\mathcal{F}$

Black-Box-α inference simplifies the approximate posterior $q(\mathbf{u})$ of Power EP, a generalisation of EP, by tying its factors. Furthermore, one can recover the variational objective for training DGPs proposed by Hensman and Lawrence (2014), by taking the limit $\alpha \to 0$ in (2.12) using L'Hopital's rule:

$$E_{VB} = \mathbb{E}_{q(\mathbf{u})}[\log p(\mathbf{u}, \mathbf{x}_{1:N}, y_{1:N}) - \log q(\mathbf{u})] \tag{2.13}$$

The DGP training procedure presented in Bui et al. (2015) runs Stochastic EP algorithm to approximate the true posterior $p(\mathbf{u}|\mathbf{x}_{1:N}, y_{1:N})$ using message passing updates on the tied factor $t(\mathbf{u})$, Li et al. (2015), to later compute the approximate marginal log-likelihood $\mathcal{F}$ used for hyper-parameter tuning. Later Bui et al. (2016) obtained an improved performance in DGP training by employing direct optimisation on $\mathcal{F}$ in (2.10) for jointly tuning of hyperparameters and inducing inputs, as well as, approximating factors, taking advantage of the tied factor constraint.

The stationary conditions for SEP-energy derived in Hernández-Lobato et al. (2015) imply that both optimisation procedures yield an approximate posterior $q(\mathbf{u})$ where moments are matched with the mean of the $p(y_i|\mathbf{x}_i, \mathbf{u})q^{\setminus 1}(\mathbf{u})$ moments. In addition, the moment matching step in SEP-based training, where local Kullback–Leibler divergences $\mathcal{KL}(p(y_i|\mathbf{u}, \mathbf{x}_i)q^{\setminus 1}(\mathbf{u})||t(\mathbf{u})q^{\setminus 1}(\mathbf{u}))$ are minimised, requires evaluating $\log \mathcal{Z}_i = \log \int_{\mathbf{u}} p(y_i|\mathbf{x}_i, \mathbf{u})q^{\setminus 1}(\mathbf{u})d\mathbf{u}$ and its gradients with respect to mean and covariance of the cavity $q^{\setminus 1}(\mathbf{u})$. Hence both training procedures rely on the evaluation $\log \mathcal{Z}_i$ and its gradients, which is analytically intractable for $L > 1$ as the data likelihoods given $\mathbf{u}$ are non-linear and the propagation of the cavity through the DGP network yields a complex distribution.

2.2.3 Probabilistic Back-propagation

First, we present an algorithm equivalent to Assumed Density Filtering (ADF), Li et al. (2015), that allows us to efficiently propagate Gaussians forwards through DGP layers projecting the resulting complex distributions into moment matched Gaussians. Then, a similar Probabilistic Back-propagation scheme to Adams (2015) can be used to obtain estimates $\log \mathcal{Z}_i$ and its gradients.

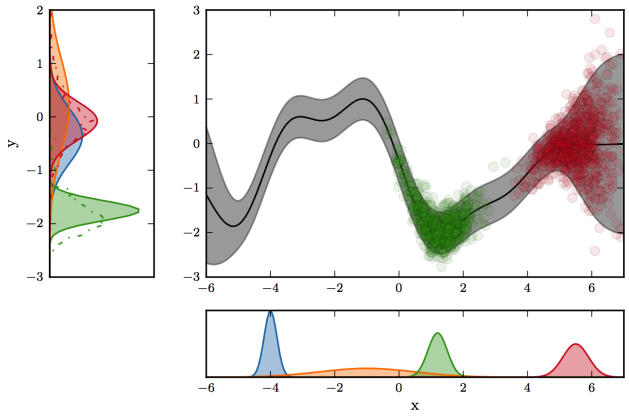


Figure 2.4 Forward propagation of Gaussian factors through GP-layer

This method obtains a Gaussian approximation to $\mathcal{Z}$ in a sequential fashion, where for each GP layer l we compute Gaussian approximations $q(h^{(l)})$ with the following recursive relation:

$$q(h^{(l)}) = \int_{h^{(l-1)}} q(h^{(l)}|h^{(l-1)})q(h^{(l-1)})dh^{(l-1)} \tag{2.14}$$

$$\text{where } q(h^{(l)}|h^{(l-1)}) = \int_{u^{(l)}} p(h^{(l)}|h^{(l-1)}, u^{(l)})q^{\setminus 1}(u^{(l)})du^{(l)} \tag{2.15}$$

with $q(h^{(1)}) = \int_{u^{(1)}} p(h^{(1)}|\mathbf{x}, u^{(1)})q^{\setminus 1}(u^{(1)})du^{(1)}$ and $q(h^{(L)}) = q(y) = \mathcal{Z}$. Contrary to the approach in Damianou and Lawrence (2013), we can exactly marginalise out the inducing outputs $u^{(l)}$ of each layer to obtain Gaussian conditional distributions $q(h^{(l)}|h^{(l-1)}) = \mathcal{N}(h^{(l)}; m_{l|l-1}, v_{l|l-1})$ with:

$$m_{l|l-1} = \mathbf{K}_{hu}^{(l)}\mathbf{K}_{uu}^{(l)-1}\mathbf{m}_l^{\setminus 1} \tag{2.16}$$

$$v_{l|l-1} = \mathcal{K}^{(l)}(h^{(l-1)}, h^{(l-1)}) + \sigma_{(l)}^2 - \mathbf{K}_{hu}^{(l)}\mathbf{K}_{uu}^{(l)-1}\mathbf{K}_{uh}^{(l)} + \mathbf{K}_{hu}^{(l)}\mathbf{K}_{uu}^{(l)-1}\mathbf{V}_l^{\setminus 1}\mathbf{K}_{uu}^{(l)-1}\mathbf{K}_{uh}^{(l)} \tag{2.17}$$

$$\text{and } q^{\setminus 1}(u^{(l)}) = \mathcal{N}(u^{(l)}; \mathbf{m}_l^{\setminus 1}, \mathbf{V}_l^{\setminus 1}) \tag{2.18}$$

Note $q(h^{(1)})$ is also Gaussian following the above. As a final step, we use the law of iterated conditionals to estimate the integral (2.14) and complete the Gaussian approximation of $q(h^{(l)}) \approx \mathcal{N}(h^{(l)}; m_l, v_l)$ through moment matching:

$$m_l = \mathbb{E}_{q(h^{(l-1)})}[m_{l|l-1}] = \mathbb{E}_{q(h^{(l-1)})}[\mathbf{K}_{hu}^{(l)}]\mathbf{A}^{(l)} \tag{2.19}$$

$$\begin{aligned} v_l &= \mathbb{E}_{q(h^{(l-1)})}[v_{l|l-1}] + Cov_{q(h^{(l-1)})}[m_{l|l-1}] \\ &= \mathbb{E}_{q(h^{(l-1)})}[\mathcal{K}^{(l)}(h^{(l-1)}, h^{(l-1)})] + \sigma_{(l)}^2 - m_l^2 + tr(\mathbf{B}^{(l)}\mathbb{E}_{q(h^{(l-1)})}[\mathbf{K}_{uh}^{(l)}\mathbf{K}_{hu}^{(l)}]) \end{aligned} \tag{2.20}$$

where $\mathbf{A}^{(l)} = \mathbf{K}_{uu}^{(l)-1}\mathbf{m}_l^{\setminus 1}$ and $\mathbf{B}^{(l)} = \mathbf{K}_{uu}^{(l)-1}(\mathbf{V}_l^{\setminus 1} + \mathbf{m}_l^{\setminus 1}\mathbf{m}_l^{\setminus 1T})\mathbf{K}_{uu}^{(l)-1} - \mathbf{K}_{uu}^{(l)-1}$ which are common to all data points. It is worth noticing that the expectations taken in (2.19) and (2.20) are only analytically tractable for specific kernel functions such as the SE kernels used in this project. Then, in the forward propagation step, we compute Gaussian approximations of $q(h^{(1)}), \cdots, q(h^{(L-1)})$ storing the gradients of m_l and v_l with respect to layer hyperparameters, pseudo-inputs and m_{l-1} and v_{l-1}. In the backwards step, we compute $\log \mathcal{Z} = \log \mathcal{N}(y; m_L, v_L)$

and use the chain-rule to take gradients through the ADF procedure.

The `geepee` library uses a minibatch approximation of $\mathcal{F}$ for stochastic optimisation, which gives an unbiased noisy estimate of the objective. The minibatch size can play a fundamental roll in the convergence of the objective during training as we will see later on in future chapters.

2.3 Evaluation metrics

In future experiments, we will use the three metrics described below to evaluate the predictive performance of our models on the test set and compare them. Given[2] a training set $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{N_{train}}$ and a test set $\{\hat{\mathbf{x}}_j, \hat{y}_j\}_{j=1}^{N_{test}}$, we denote the predictive distribution as $p(\hat{y}|\hat{\mathbf{x}}, \mathcal{D}, \alpha)$ with mean $\mu_{pred}(\hat{\mathbf{x}})$ and variance $\sigma^2_{pred}(\hat{\mathbf{x}})$.

- **Mean Squared Error (MSE):**

$$\text{MSE} = \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} |\hat{y}_j - \mu_{pred}(\hat{\mathbf{x}}_j)|^2 \tag{2.21}$$

- **Mean Negative Log-Predictive Probability (MNLPP):**

$$\text{MNLPP} = -\frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \log p(\hat{y}_j|\hat{\mathbf{x}}_j, \{\mathbf{x}_i, y_i\}_{i=1}^{N_{train}}, \alpha) \tag{2.22}$$

$$= \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} \frac{1}{2} \log(2\pi\sigma^2_{pred}(\hat{\mathbf{x}}_j)) + \frac{|\hat{y}_j - \mu_{pred}(\hat{\mathbf{x}}_j)|^2}{2\sigma^2_{pred}(\hat{\mathbf{x}}_j)} \tag{2.23}$$

- **Sample based Negative Log-Predictive Probability:**
  Let $f^{(m)} \sim p(f|\mathcal{D}, \alpha)$ for $m = 1, \cdots, M$ be M samples generated from the posterior of a fully trained model. Assume we model observations $\hat{y}$ given a sample $f^{(m)}$ as $p(\hat{y}|f^{(m)}, \hat{\mathbf{x}}) = \mathcal{N}(\hat{y}; f^{(m)}(\hat{\mathbf{x}}), \sigma^2_{out})$. Then, the predictive distribution can be estimated with a mixture of M Gaussians centred at samples $f^{(m)}$:

$$p(\hat{y}|\hat{\mathbf{x}}, \mathcal{D}, \alpha) = \int p(\hat{y}|f, \hat{\mathbf{x}}) p(f|\mathcal{D}, \alpha) df \tag{2.24}$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} \mathcal{N}(\hat{y}; f^{(m)}(\hat{\mathbf{x}}), \sigma^2_{out}) := \hat{p}_M(\hat{y}|\hat{\mathbf{x}}, \mathcal{D}, \alpha) \tag{2.25}$$

  The sample-based MNLPP score is obtained simply by substituting $\hat{p}_M(\hat{y}|\hat{\mathbf{x}}, \mathcal{D}, \alpha)$ into (2.22).

In hold-out test sets, where the training and test sets have a similar distribution, the sample-based and regular MNLPP return the same scores. However, in out-of-sample test sets with points isolated from the training set, the sample-based MNLPP score gives a better measure of the predictive ability of a sparse Deep Gaussian Process, as it ignores the poor moment matching approximation made at these isolated points.

[2] Although, described for 1-dimensional output regression ($y \in \mathbb{R}$), it can be easily generalised to higher dimensional outputs

Chapter 3

Experiments with toy data

In this chapter, we evaluate the representational capacity of sparse DGPs in two toy datasets traditionally challenging for Squared Exponential (SE) GPs, namely, the step function and a piecewise linear function. Our goal is to demonstrate that despite the hierarchical structure of DGPs, the sparsification of the GP-layers can damage predictive performance. We report the anomalies exhibited by DGPs when compared to full GPs and perform experiments on out-of-sample test sets.

3.1 Step function

In this section, we perform experiments on a 100 point dataset generated from a noisy step function similar to the one in Rasmussen and Williams (2004). $g(x_i) = +1 + \epsilon_i$ if $x > 0$ and $g(x_i) = -1 + \epsilon_i$ otherwise, where $\epsilon_i \sim \mathcal{N}(0, 0.025)$ is a homoscedastic Gaussian noise. Despite its simplicity, the step function has become a benchmark toy dataset for DGPs as SE kernel GPs struggle to model its discontinuity at zero.Rasmussen and Williams (2004) showed that a function drawn from a SE-GP would have infinite support – infinite continuous derivatives – everywhere with probability 1. In addition, its first derivative is point-wise distributed as $\mathcal{N}(0, v^2_{(l)}/l^2_{(l)})$, requiring a very short lengthscale to model the jump at zero.

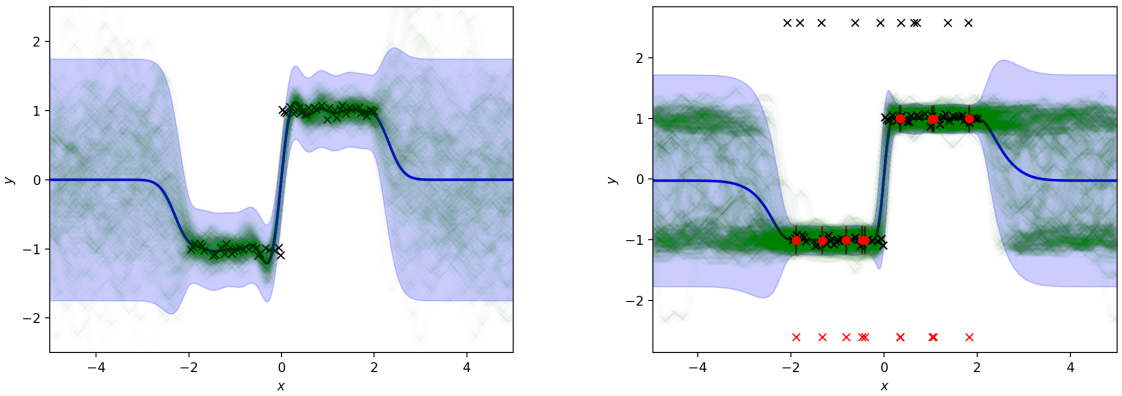
3.1.1 Experiments

Scores in table 3.1 are obtained using training set of 60 equally spaced training points and a hold-out test set of 40. The SE kernel AEP-DGP architectures are trained by directly optimising the objective $\mathcal{F}$ in Chapter 2 for hyperparameters and 10 pseudo-point, using Adam with an appropriate learn rate (0.01), no minibatch, 1,500 iterations and the default `geepee` initialisation[1].

[1]The default `geepee` initialisation is described in full detail in Chapter 5

Model	Hidden-size	MSE	MNLPP
Full GP	–	0.009	-0.315
DGP	[2]	0.003	-0.845
	[1,1]	0.001	-1.058
	[3,2]	0.001	-1.055
	[3,2,2]	0.001	-1.036

Table 3.1 Step function scores as hidden-layer configurations changes.



(a) Full GP model.

(b) AEP-DGP model with $M = 10$.

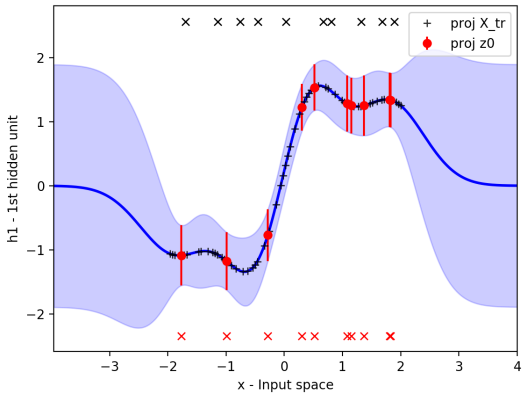
Figure 3.1 Step function regression and samples from GP and AEP-DGP.

Results demonstrate that a single hidden-layer[2] gives AEP-DGP sufficient representational power to overcome full GP scores. Adding an extra hidden layer decreases the MSE and MNLPP scores. However, as we go deeper this is no longer the case as the AEP-DGP starts to over-fit the small training set. It is worth pointing out that AEP-DGPs have a tendency to not employ their full representational capacity by only using one hidden unit per layer, "shutting down" – explaining data simply with a large pseudo-point noise – additional units as it is shown in figure 3.2b. This shallow behaviour is also observed in Damianou and Lawrence (2013) variational DGPs.

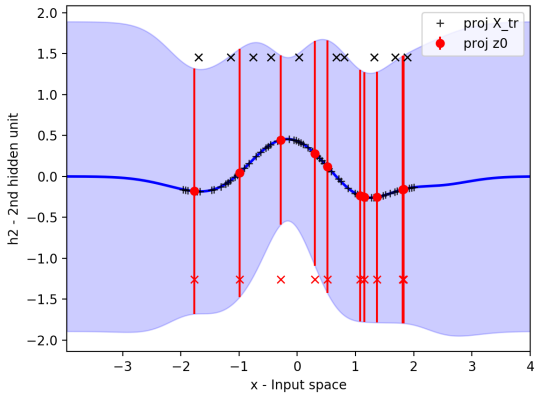
Each of the figures 3.2 show the training points (black crosses) different GP-mappings $f^{(l)}$ of an optimised AEP-DGP for the step function with: mean functions in dark blue with 2σ-error-bars (light blue), pseudo-outputs $u^{(l)}$ (red dots) and their standard deviation (vertical red lines). The black and red x-s represent the initial and optimised location of $z^{(1)}$ pseudo-inputs.

Similarly, figure 3.1, shows the mean function (dark blue) with 2σ-error-bars (light blue) and 100 samples (green) from a full GP (left) and one-hidden-layer AEP-DGP (right) with two hidden units and 10 inducing points (red). We observe how the outside the training points the full GP model quickly returns to the zero prior mean with a lot of uncertainty. In contrast, outside this region the samples from the AEP-DGP show the bimodal behaviour as a consequence the

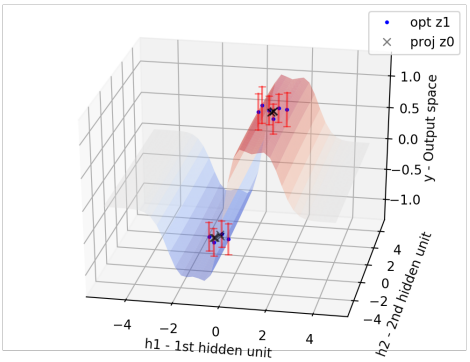
[2] [2] represents a AEP-DGP with a hidden-layer with two hidden units. Similarly, [3,2] represents a AEP-DGP with two hidden-layers with 3 hidden units in the first layer and 2 in the second one



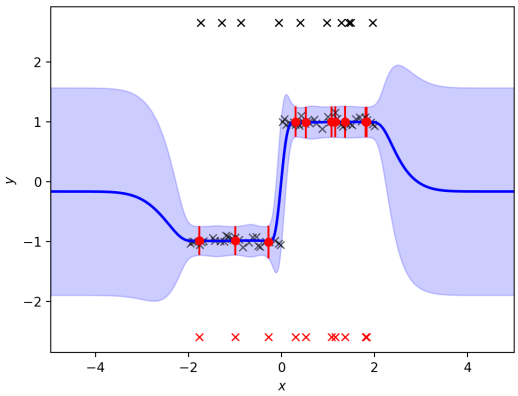
(a) $f_1^{(1)}(\mathbf{x})$ GP-mapping.



(b) $f_2^{(1)}(\mathbf{x})$ GP-mapping.



(c) $f^{(2)}(h^{(1)})$ GP-mapping.



(d) AEP-DGP regression from input to output space.

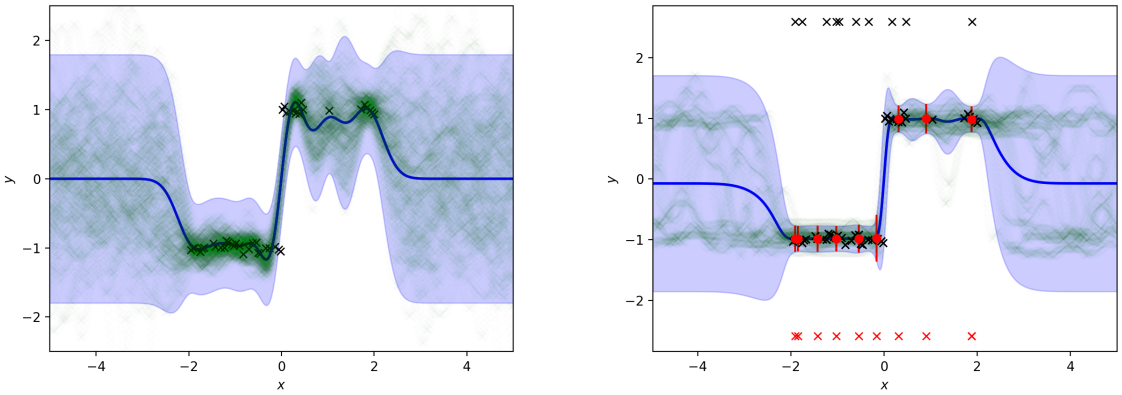
Figure 3.2 Representation of $f^{(l)}$ GP-mappings and AEP-regression plots for step function.

hierarchical structure capturing this extreme non-stationarity. This can also be appreciated at the discontinuity at zero, where SE kernel puts zero mass , and the GP uses a short lengthscale and smoothly transitions from the bottom to the top part, whereas the AEP-DGP is capable of modelling the sharp jump shrinking with the composition of two short lengthscale GPs (see figure 3.2a, c).

3.1.2 Out-of-sample test sets

Our interest for out-of-sample test sets is motivated by the isolated points frequently found on highly dimensional black-box functions in Bayesian Optimisation. To recreate similar conditions, we remove interval blocks from the training set, leaving a few points in between. Then , we select the test set to be in the region where the interval blocks were removed.

In this case, we remove a large portion of the top part of the step function leaving a single point in between. As shown in figure 3.3, the full GP model attempts to return to the prior increasing the uncertainty in that region significantly. On the contrary, AEP-DGP model ignores the



(a) Full GP model. (b) AEP-DGP model with $M = 10$.

Figure 3.3 Out-of-sample full GP and AEP-DGP regression on step function.

Model	MSE	MNLPP	Sample MNLPP
Full GP	0.040	0.549	–
DGP	0.003	-0.648	-0.832

Table 3.2 Out-of-sample test scores for step function.

initialisation and locates one of the 10 inducing points at the middle point. Scores in table 3.2 reserve as justification of the previous explanation. Important to note that the bimodal behaviour of the AEP-DGP is still present here, showing the robustness of AEP-DGP training point removal.

3.2 Piecewise linear function

We build on the characteristic properties of the step function – discontinuity at zero and piecewise linearity – to design a more challenging function that would show the potential damage of sparsification on the representational capacity of AEP-DGPs, while also force SE-GPs to choose short lengthscales to model the jumps. In this section, we perform experiments on a 100 point dataset generated by adding homoscedastic Gaussian noise $\epsilon_i \sim \mathcal{N}(0, 0.025)$ to the function:

$$disc(x) = \begin{cases} -x & \text{if } x < -1.75 \\ x + 2.5 & \text{if } -1.75 < x < -1 \\ -0.25x & \text{if } -1 < x < 0.5 \\ 0.25x + 2 & \text{if } x > 0.5 \end{cases}$$

3.2.1 Experiments

Scores in table 3.3 are obtained using training set of 60 equally spaced training points and a hold-out test set of 40. For these experiments, we vary the number of inducing points in the

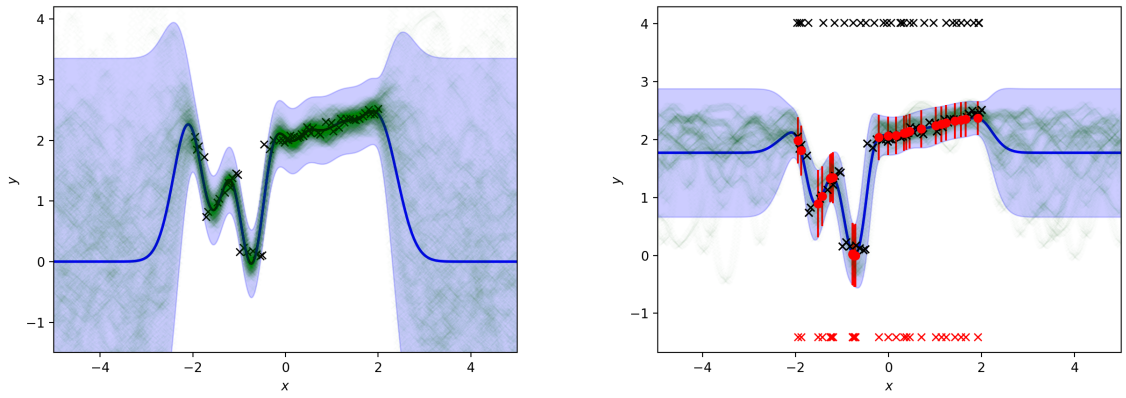
Model	M	MSE	MNLPP	σ^2_{out}
Full GP	–	0.033	-0.147	0.16
DGP	10	0.042	0.189	0.13
	20	0.032	0.019	0.11
	30	0.026	-0.238	0.08

Table 3.3 $disc(\cdot)$ scores as hidden-layer configurations changes.

AEP-DGP model while maintaining an architecture of a single hidden-layer with two units.

The optimiser configuration is exactly the same as in the previous experiments. Figure 3.4 shows that since the output of the $disc(\cdot)$ function is not normalised, outside the training set region the uncertainty of AEP-DGP is significantly lower than the full GPs that quickly return to the prior zero mean. This can also be observed in the generated samples (green) in figure 3.4b that demonstrate that AEP-DGP is capable of learning some kind of periodic function representation.

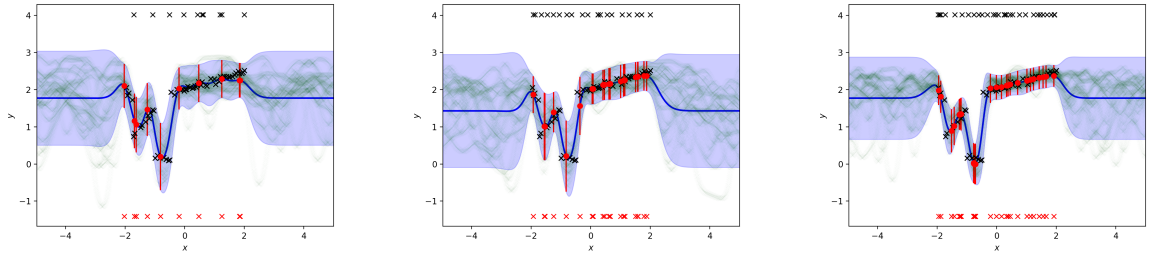
In table 3.3, we see that it takes about 30 inducing points – half of the training size – for the AEP-DGP model to overcome the full GP. In figures 3.5a,b,c we observe pseudo-points clumping instead of spreading out evenly in the region $[-1, -0.5]$, affecting its predictive performance. This behaviour was also noticed by Bauer et al. (2016) in standard FITC sparse GPs, where the optimiser finds solutions that clump pseudo-points on the top of another to avoid the complexity penalty of the FITC objective. One could argue that AEP-DGP architectures with few hidden layers are not sufficient to repair some of damage caused by the sparsification of the GP-mappings. We will look more in depth into the pathologies the modelling capacity of this type of architectures in Chapter 4.



(a) Full GP model.

(b) AEP-DGP model with $M = 30$.

Figure 3.4 Regression and samples from GP and AEP-DGP on $disc(\cdot)$ function dataset.



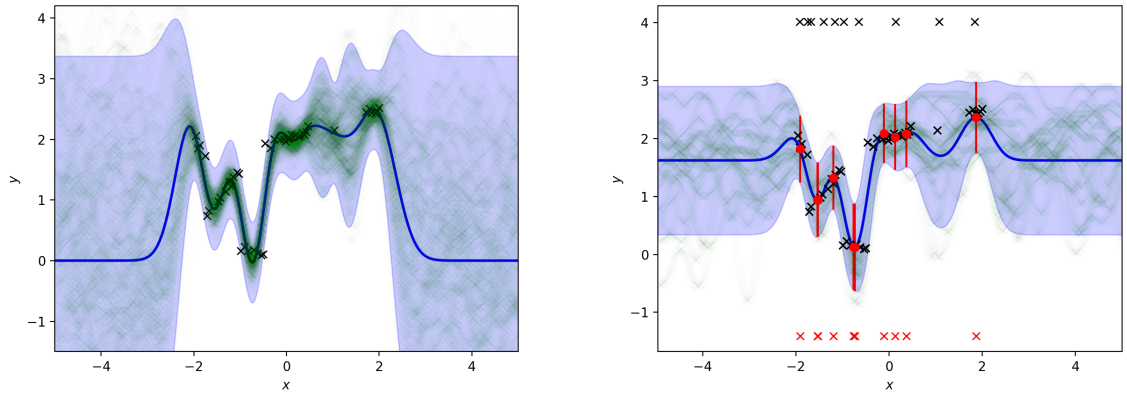
(a) M=10. (b) M=20. (c) M=30.

Figure 3.5 Regression and samples from AEP-DGP as the number inducing points M increases.

Model	M	MSE	MNLPP	Sample MNLPP
Full GP	–	0.040	0.380	–
DGP	10	0.203	0.677	0.563
	20	0.048	0.220	0.160

Table 3.4 Out-sample test scores for $disc(\cdot)$ function.

3.2.2 Out-of-sample experiments



(a) Full GP model. (b) AEP-DGP model with $M = 10$.

Figure 3.6 Out-of-sample full GP and AEP-DGP regression on $disc(\cdot)$ function.

In this case, we remove a large portion of the right part of the $disc(\cdot)$ function leaving a single point in between. As shown in figure 3.6, the full GP model increases the uncertainty in that region when attempting to return to the prior, but the predictive mean still gives good predictions in that region. On the contrary, the gradients of AEP-DGP model move the 10 inducing points away from the middle point ignoring the initialisation (black crosses on the top). Scores in table 3.4 show how at least 20 inducing points are need to beat full GPs, at least in terms of MNLPP.

3.3 Conclusions

In this chapter, we tested the modelling ability of AEP-DGPs on two benchmark toy datasets, where the predictive performance of standard SE-GPs is known to be limited. We observe that AEP-DGPs have a tendency to find configurations which do not make use of their full representational capacity, either by having a single active unit per layer or by clumping its pseudo-points.

Motivated by this indication that the sparsification of the GP-mappings can have a damaging impact on the predictive performance of one-hidden-layer AEP-DGPs, in the next chapter we test these models on medium-size datasets formed by samples generated from a DGP with the same architecture.

Chapter 4

Experiments with DGP samples

In this chapter, we investigate the performance of AEP-DGPs on a medium-size synthetic dataset formed by 10 samples from a one-hidden-layer DGP. We first explain how to sample from a DGP and how the dataset for our experiments was generated. Then, we analyse the pathological behaviour exhibited by AEP-DGPs when compared to sparse and full GPs.

4.1 Generating samples from a Deep Gaussian Process

One can understand a Deep Gaussian Process as generative model based on the composition of a series of GP mappings. We take a one-hidden-layer DGP with one-dimensional inputs and outputs as running example (see figure 5.6). We can generalise the generative process to higher-dimensional hidden-layers simply by generating multiple i.d.d. samples from a one-dimensional-output GP mapping as demonstrated in Damianou and Lawrence (2013).

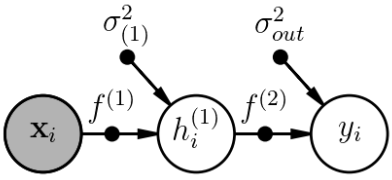


Figure 4.1 Graphical model of one-hidden-layer Deep Gaussian Process.

Suppose we are given a finite set of input locations $\{\mathbf{x}_i\}_{i=1}^N$ and GP mappings $f^{(l)} \sim \mathcal{GP}(0, \mathcal{K}^{(l)})$, characterised by the kernel functions $\mathcal{K}^{(l)}(\cdot,\cdot) : \mathbb{R}^{D_{l-1}} \times \mathbb{R}^{D_{l-1}} \mapsto \mathbb{R}$. As described in Chapter 2, we can generate hidden-layer samples by computing the gram covariance matrix $(\mathbf{K}^{(1)}_{hh})_{ij} = \mathcal{K}^{(1)}(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j \in \{1, \cdots, N\}$ and sampling from the multivariate Gaussian:

$$h_i^{(1)} \sim \mathcal{N}(0, \mathbf{K}^{(1)}_{hh} + \sigma^2_{(1)} I) \tag{4.1}$$

Similarly, to generate outputs $y_i = f^{(2)}(h_i^{(1)}) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma_{out}^2)$ from the hidden-layer values, we simply compute the gram covariance matrix $(\mathbf{K}_{hh}^{(2)})_{ij} = \mathcal{K}^{(2)}(h_i^{(1)}, h_j^{(1)})$ for $i, j \in \{1, \cdots, N\}$ and sample from the multivariate Gaussian:

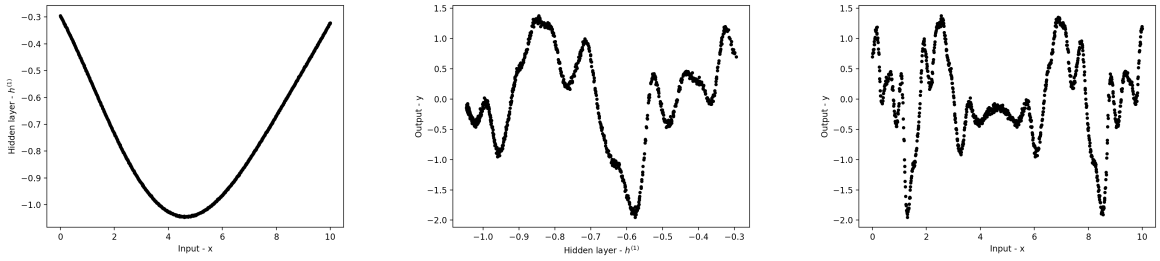
$$y_i \sim \mathcal{N}(0, \mathbf{K}_{hh}^{(2)} + \sigma_{out}^2 I) \tag{4.2}$$

The samples $\{y_i\}_{i=1}^N$ obtained correspond to samples from a one-hidden-layer Deep Gaussian Process at input locations $\{\mathbf{x}_i\}_{i=1}^N$.

In this project, we will restrict ourselves to GP mappings with squared exponential (SE) kernel functions:

$$\mathcal{K}^{(l)}(\mathbf{x}, \mathbf{x}') = v_{(l)}^2 \exp(-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2l_{(l)}^2}) \tag{4.3}$$

The values taken by a kernel function give us a measure of the similarity between two points. Hence the shape of the sampled function is controlled by: the characteristic lengthscale $l_{(l)}$, which controls the smoothness, and $v_{(l)}^2$, which controls the variance or range of the function as pointed in Rasmussen and Williams (2004).



(a) $f^{(1)}(\mathbf{x})$. (b) $f^{(2)}(h^{(1)})$. (c) $f^{(1:2)}(\mathbf{x})$.

Figure 4.2 (a)(b) Functions drawn from two GP mappings and their composition in (c).

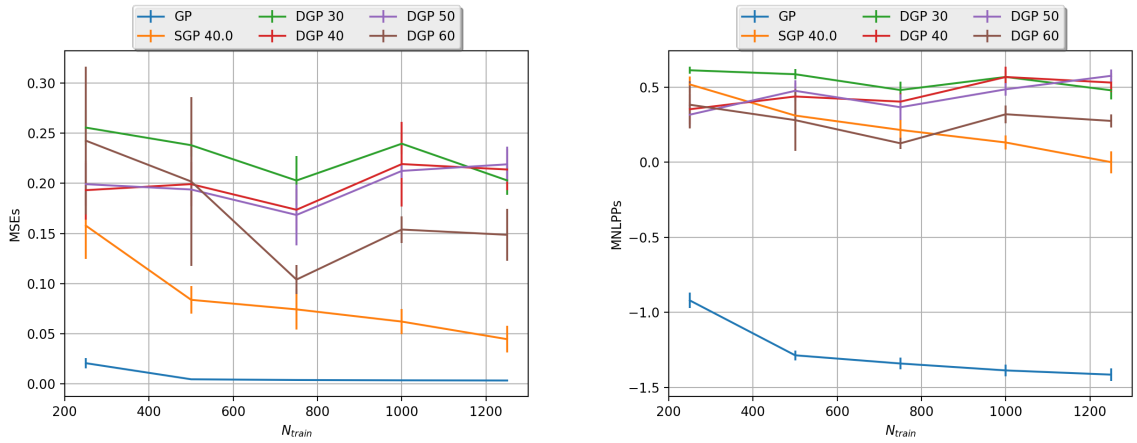
4.2 Experiments on DGP samples

In this section test the predictive performance of AEP-DGPs on a medium-size one-dimensional synthetic dataset formed by 10 samples generated from a one-hidden-layer DGP as described above. Our interest in this type of dataset relies on the non-stationary behaviour of the samples with length-scales varying across input space, a property present in challenging higher-dimensional problems in Bayesian Optimisation.

Samples are generated at 2500 equally-spaced input points on $[0, 10]$ by first sampling from a large length-scale – relative to interval length – bottom layer GP (for example: $l_{(1)} = 5$) and then from a short length-scale top layer GP (for example: $l_{(2)} = 0.025$). Variances are set to be the same $v_{(1)} = v_{(2)} = 1$ and the noise parameters $\sigma_{(1)}^2$ and σ_{out}^2 are kept small (of order 10^{-4}).

Scores in figure 4.3 are obtained by testing our models on a hold-out test set of a 1000 sample points, as the training set size increases from 250 to 1250 points. Our AEP-DGP model consisting of a single hidden-layer with two hidden units is trained by directly optimising the objective $\mathcal{F}$ in Chapter 2 for hyperparameters and pseudo-points. For the scores in figure 4.3, we used a default configuration of the optimiser Adam with appropriate learn rate (0.01), a mini-batch size of 250 points, the default `geepee` initialisation and no fixed hyperparameters running for 1,500 iterations. However, it should be pointed out that due to the high variance of the initial AEP-DGP scores, experiments were repeated using alternative configurations and only the best performing were selected to form the graphs in figure 4.3. Some of these configurations include: using alternative optimization schemes such as L-BFGS-B, chunking – optimising subsets of hyperparameters iteratively – or CG; fixing the output noise σ^2_{out} and pseudo-input locations $z^{(1)}$; increasing the number of iterations and changing the learning rate and mini-batch sizes. None of them were particularly successful, with the default configuration giving close-to-best scores most of the time.

Results in figure 4.3 are presented using the mean scores and σ-error bars over 10 DGP samples. Before the experiments, we expected AEP-DGPs to outperform full GPs and VFE sparse GPs given the same number of inducing points M in large enough datasets with $N_{tr} > 750$ say, improving scores as N_{tr} increased. However, the experiment results contradicted our intuition. Full GPs are the best performing model and VFE sparse GPs with $M \geq 40$ outperform AEP-DGPs for all M values up to 60. In addition, AEP-DGPs show an oscillatory trend, not necessarily improving scores as N_{tr} or M increases. This brings to question whether AEP-DGP suffer from sparsity problems or the underlying issue are the optimisation dynamics.



(a) MSE scores.

(b) MNLPP scores.

Figure 4.3 Scores of GP, VFE sparse GP with $M = 40$ and AEP-DGP with $M \in \{30, 40, 50, 60\}$ as N_{tr} increases.

4.2.1 Pathological behaviour

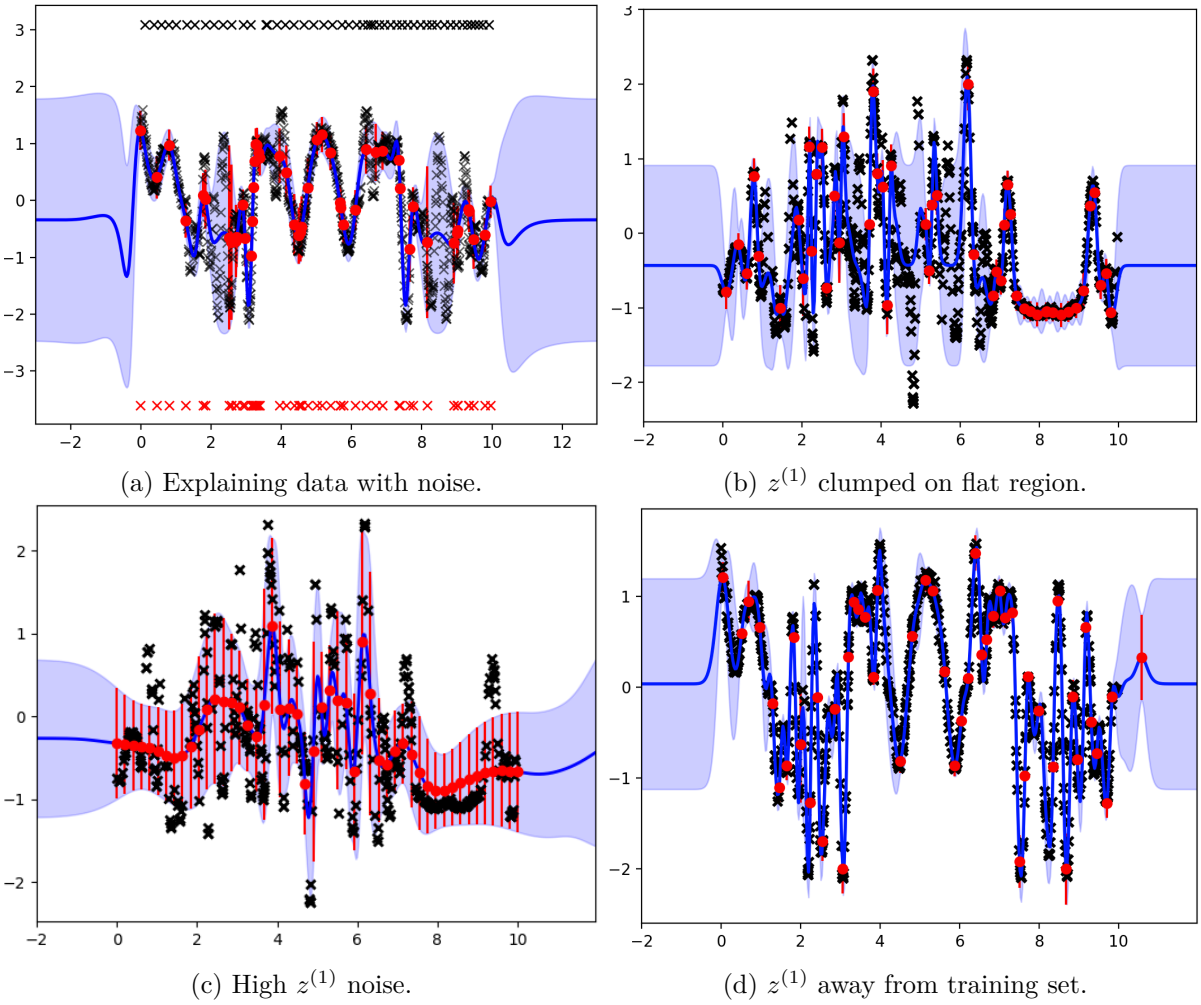
In figure 4.4 we show examples of the pathological behaviour exhibited by AEP-DGPs during the collection of scores in figure 4.3. We divide these pathologies in four groups:

- **Explaining data with noise:** As shown in figure 4.4a, in certain occasions the optimisation procedure finds AEP-DGP configurations that explain the data by increasing the output noise σ^2_{out}. This shows that even with sufficient inducing points, the AEP-DGP model can not capture the short length-scales well, giving up in those regions. Our attempts to fix the output noise σ^2_{out} during training resulted in a very slow convergence of the objective yielding poor results.
- **Clumped $z^{(1)}$:** As shown in figure 4.4a,b sometimes inducing points are clumped on the top of each other, resulting on a poor distribution of $z^{(1)}$-s that damages the predictive performance of AEP-DGPs. This could serve as an explanation for the small impact of increasing the number of inducing points observed in figure 4.3. A similar behaviour was reported by Bauer et al. (2016) in standard FITC sparse GPs. Inducing inputs were clumped together in order to reach a local minima by reducing the complexity penalty associated with having more pseudo-points in the FITC objective.
- **High $z^{(1)}$ noise:** As shown in figure 4.4c, instead of explaining data by increasing the output noise σ^2_{out}, here the posterior variance of the bottom layer inducing points $S_u^{(1)}$ is increased. This results in a very poor predictive performance, where many training points are left out of the confidence regions. Interestingly, FITC sparse GPs have a tendency of underestimate noise at inducing points due to the heteroscedastic noise induced by the FITC objective as pointed in Snelson and Ghahramani (2006).
- **$z^{(1)}$ away from training set:** As shown in figure 4.4d, sometimes the optimiser finds local optima in configurations that place inducing points away from the training set, where the uncertainty is large. We also observe these "outliers" with FITC sparse GPs in practice.

Despite of our efforts, these pathologies are recurrent issue across all our experiments on DGP samples, independently of the dataset or the optimiser configuration used.

4.3 Conclusions

After having a closer look at counter-intuitive results obtained by AEP-DGPs on samples from DGPs, we observed some pathologies that constrain the representational capacity of AEP-DGPs. These pathologies seem to be closely related to issues in the optimisation dynamics as they occur independently of the number of inducing points used. Thus, we conclude that the optimisation procedure requires a more careful treatment.



(a) Explaining data with noise.

(b) $z^{(1)}$ clumped on flat region.

(c) High $z^{(1)}$ noise.

(d) $z^{(1)}$ away from training set.

Figure 4.4 Examples of pathologies exhibited by AEP-DGP models.

In many optimisation problems, choosing an appropriate initialisation of the model can have a dramatic impact in the optimisation dynamics and help finding global optima. In the next chapter, we discuss a variety of initialisation schemes for AEP-DGP models and how these affect the predictive performance of AEP-DGPs.

Chapter 5

Initialisation Schemes

In this chapter, we introduce a greedy initialisation scheme for one-hidden-layer Deep Gaussian Processes with the goal of fixing the pathologies encountered in Chapters 3 and 4. We compare the performance of this and other alternative initialisation schemes to the default initialisation in the `geepee` library. Then, we analyse the impact of this initialisation in the evolution of DGP-layers during training and perform tests on reduced data domains and out-of-sample test sets.

5.1 Greedy Initialisation

In a recent review of Fully Independent the Training Conditional (FITC)[Quiñonero-candela et al. (2005), Snelson and Ghahramani (2006)] and the Variational Free Energy (VFE) [Titsias (2009)] sparse approximation methods for Gaussian Processes, Bauer et al. (2016) demonstrated that VFE has a tendency to find under-fitting solutions, mainly due to problems in the optimisation procedure. In their experiments on the 32-dimensional `pumadyn32` dataset, VFE initialised with the inducing inputs and hyperparameters of a FITC solution was the best performing sparse approximation, close to full GP performance.

Inspired by this, we propose a greedy bottom-up sparse GP initialisation of the AEP-DGP model that can be easily implemented in the `geepee` library. Although, originally intended for one-hidden-layer DGPs with Squared Exponential kernel functions, this initialisation can easily be extended to a more general form of sparse DGPs. Here, we take the one-hidden-layer AEP-DGP with two hidden units architecture used in Chapter 4 as running example.

5.1.1 Description

As described in Chapter 2, the GP-mappings of a AEP-DGP network are approximated with FITC sparse GPs at M inducing points. Before initialising our AEP-DGP model, we train and store the defining parameters of a separate sparse GP model with same number of pseudo-points M, , f_{SGP}. These defining parameters include: the characteristic length-scale l_{SGP}, the variance

v^2_{SGP}, the output noise $\sigma^2_{out,SGP}$, the induced input locations z_{SGP}, and $\theta_{1,SGP} = S_u^{-1}$ and $\theta_{2,SGP} = S_u^{-1} m_u$, the natural parameters of the posterior distribution $f_{SGP}|z_{SGP} \sim \mathcal{N}(m_u, S_u)$. The training procedure is then divided into two subsequent rounds:

- **First round:** Bottom layer GP-mappings, $f_1^{(1)}(\mathbf{x}), f_2^{(1)}(\mathbf{x})$ are initialised with the optimised sparse GP hyperparameters previously stored. Then, the AEP-DGP model is trained fixing the bottom layer hyperparameters and letting the output noise σ^2_{out} evolve. Here, we initialise the top GP-mapping, $f^{(2)}(h_1^{(1)}, h_2^{(1)})$, with the default initialisation.
- **Second round:** The entire AEP-DGP architecture is trained using the optimised model hyperparameters from the previous round. Here, the bottom layer hyperparameters are no longer fixed.

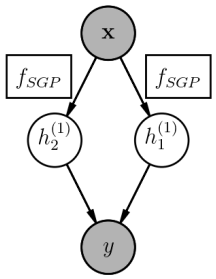


Figure 5.1 First round of training for greedy initialisation with fixed bottom layer.

A problem that intuitively arises from this greedy initialisation scheme is that the optimiser will have a tendency to find local solutions where the top-layer is highly linear, resulting in a performance similar to sparse GPs. However, by introducing a small noise in the diagonal entries of the covariance S_u, we generate an asymmetric initialisation of the hidden units, which induces a simple non-linearity in the top-layer during the first round of training that generally prevents this issue from happening.

5.1.2 Alternative initialisations

We propose alternative initialisation schemes based on the generative process behind the DGP samples used in Chapter 4 that could potentially help in the optimisation procedure. For reference we also include the default initialisation implemented in the `geepee` library:

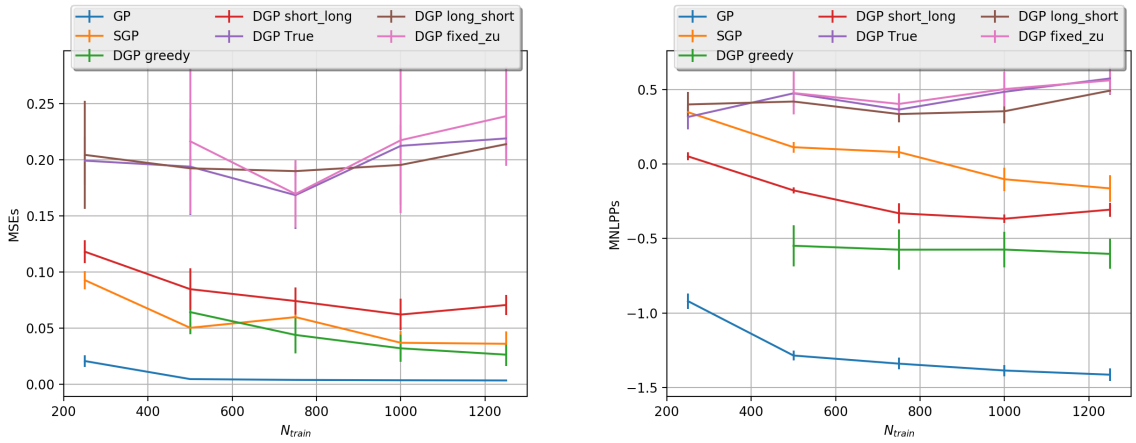
- **"Long-Short" initialisation:** The bottom layer sparse GPs are initialised with a long characteristic length-scale $l_{(1)} = 5$ – relative to the input range $[0, 10]$ of samples in Chapter 4 – and the top layer sparse GPs with a short $l_{(2)} = 0.05$. This initialises the AEP-DGP model with similar characteristic length-scales that generated the data.
- **"Short-Long" initialisation:** The bottom layer sparse GPs are initialised a short $l_{(1)} = 0.1$ – relative to the interval $[0, 10]$ – and the top layer sparse GPs with a long characteristic length-scale $l_{(2)} = 1$. This initialisation encourages configurations that distribute bottom layer inducing inputs $z^{(1)}$ across the training set, thus avoiding the clumping.
- **"Fixed $z^{(1)}$" initialisation:** Builds on the above by initialising the bottom layer inducing inputs $z^{(1)}$ on a regular grid and keeping them fixed throughout training.
- **"Default" initialisation:** The bottom GP-layer is initialised by: choosing M training points as inducing inputs $z^{(1)}$ using K-means, a characteristic length-scale[1] $l_{(1)}$ equal to the median of the distances between training points, and a variance $v^2_{(1)} = 0.25$. Following GP-layers are initialised with $z^{(l)}$ regularly spaced on the identity diagonal of the $[-1, 1]^{D_{l-1}}$ hypercube in hidden-space and $l_{(l)} = v^2_{(l)} = 1$. The natural parameters $\theta^{(l)}_1 = S_u^{-1}$ and $\theta^{(l)}_2 = S_u^{-1} m_u$ are initialised in the same way for all layers with $S_u = 0.5I$ and m_u equally spaced points in the $[-1, 1]$ interval. In addition, the output noise is initialised with a small value $\sigma^2_{out} = 0.01$.

5.2 Experiments

In this section, we analyse the impact of the initialisation scheme used on the modelling capacity of AEP-DGPs. The experiments are performed on a dataset formed by one-hidden-layer DGP samples as in Chapter 4. The default configuration of the optimiser explained in Chapter 4 is also used in this section.

Results in figure 5.2 are presented using the mean scores and σ-error bars obtained by our models over 10 DGP samples from Chapter 4. Here we compare the performance of AEP-DGPs to full GPs and VFE sparse GPs with same number of inducing points $M = 50$, as the number of training points N_{tr} increases. We initialise our AEP-DGP model using the five initialisation schemes described above. In addition, table 5.1 shows the performance AEP-DGPs depending on the initialisation used and how this relates to the initial value of the objective. Results demonstrate that our greedy initialisation outperforms the others and even improves the sparse GP scores for $N_{tr} > 500$. We also note that the default initialisation as well as "long-short" and "Fixed $z^{(1)}$" initialisations show very poor performance. The "short-long" initialisation (opposite to the generative process of the samples) obtains second-best scores outperforming the sparse GPs in terms of MNLPP.

[1] Small noise is added to most hyperparameters to avoid algebraic problems during computations.



(a) MSE scores.

(b) MNLPP scores.

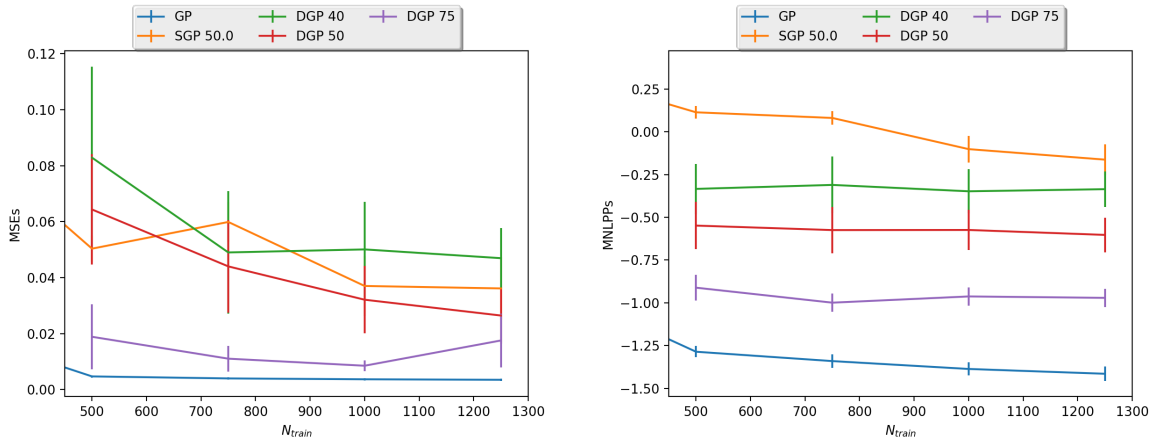
Figure 5.2 Scores of GP, VFE SGP and AEP-DGP depending on initialisations as N_{tr} increases.

Model	Initial Objective	MSE	MNLPP	σ^2_{out}
Full GP	–	0.004	-1.317	–
VFE SGP	–	0.011	-0.427	–
FITC SGP	–	0.015	-0.383	–
Greedy Round 2	0.05	**0.09**	**-0.789**	**0.09**
Greedy Round 1	4.32	0.076	-0.160	0.18
Default	12.21	0.200	0.384	0.32
Long-short	3.60	0.163	0.258	0.27
Short-long	9.17	0.068	0.219	0.15
$z^{(1)}$ fixed	40.32	0.355	0.901	0.60

Table 5.1 Ranking of initialisation schemes for dataset with $N_{tr} = 1000$, $M = 50$.

However, greedy AEP-DGPs can still not compete with full GPs on these medium-size datasets. This is explored more in depth in figure 5.3 where we compare greedy AEP-DGPs with different number of pseudo points M to full GPs. Contrary to what we observed in Chapter 4, where the AEP-DGPs showed an oscillatory behaviour as M increases, here greedy AEP-DGPs improve their performance. However, this improvement is limited and highly dependent on the performance of the sparse GP used in the greedy initialisation.

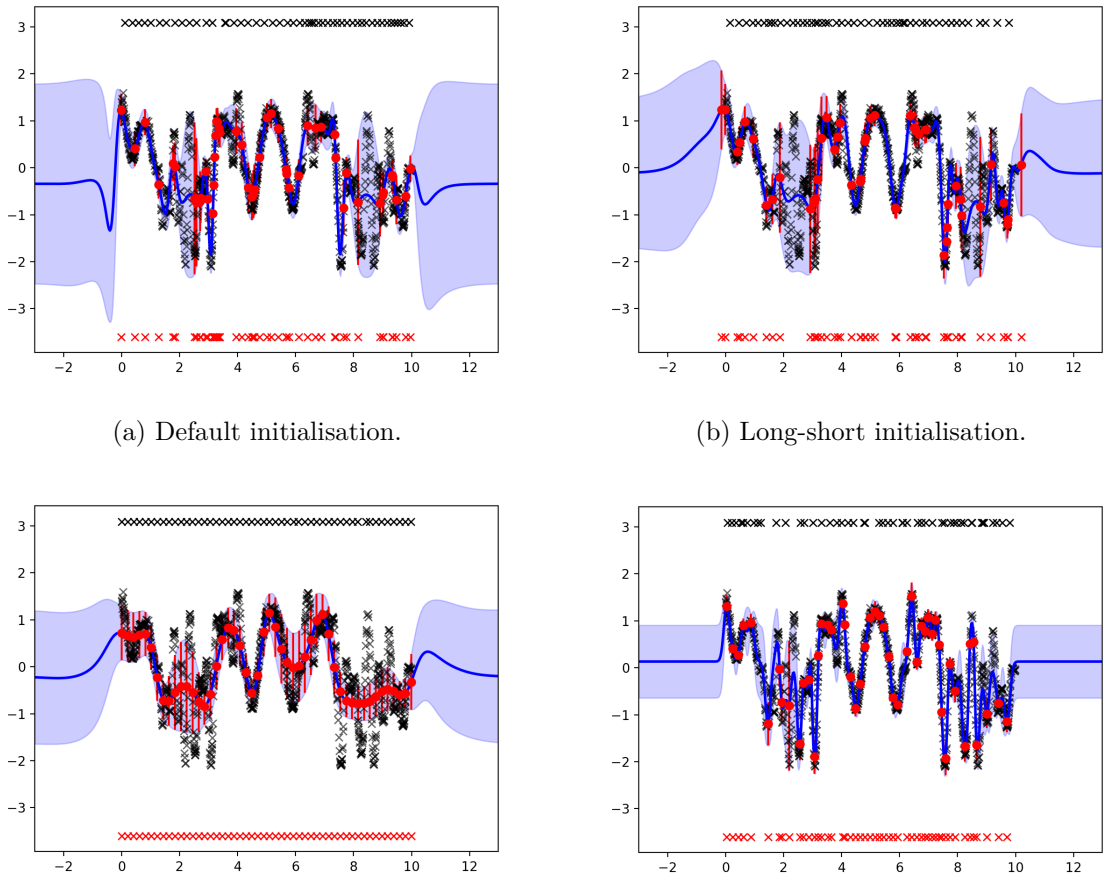
Figure 5.4 shows how AEP-DGP predictions using these initialisations exhibit the pathologies described in Chapter 4. In particular, figures 5.4a,b show that in the default and "long-short" initialisations there are some regions where data is explained by noise whilst others have inducing points clumped. The "long-short" initialised AEP-DGP also presents two outlier pseudo-points, located away from the train set, with a high variance. In figure 5.4c, the "fixed $z^{(1)}$" initialised AEP-DGP uses a very high variance at the pseudo-points to explain the data. This results in very poor predictions as several training points are left out of the confidence regions, which also happens in figure 5.4d with "short-long" initialised AEP-DGP.



(a) MSE scores.

(b) MNLPP scores.

Figure 5.3 Scores of GP, VFE sparse GP and AEP-DGP greedy initialisation with $M \in \{30, 40, 50, 60\}$ as N_{tr} increases.



(a) Default initialisation.

(b) Long-short initialisation.

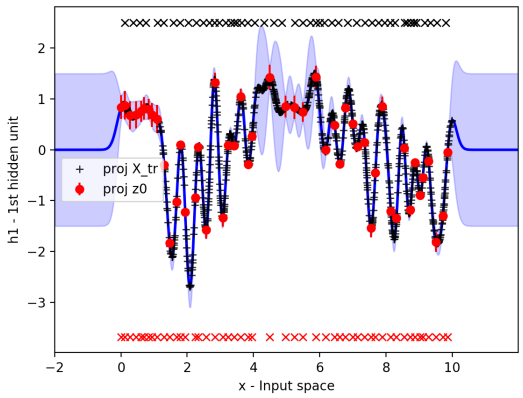
(c) Fixed $z^{(1)}$ initialisation.

(d) Short-long initialisation.

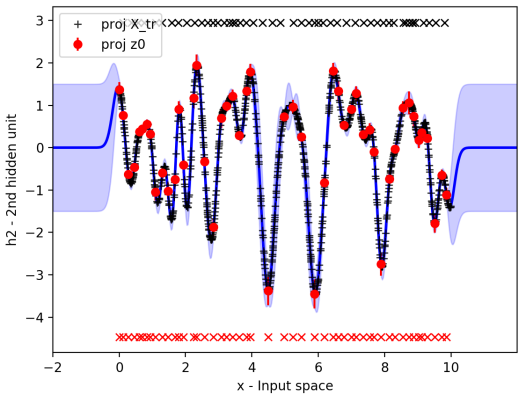
Figure 5.4 Examples of AEP-DGP behaviour depending on initialisation.

Figure 5.5 shows the optimised GP-mappings after a greedy initialisation of a AEP-DGP model with $M = 50$ inducing points. In figure 5.5d, the pathological behaviour described in Chapter

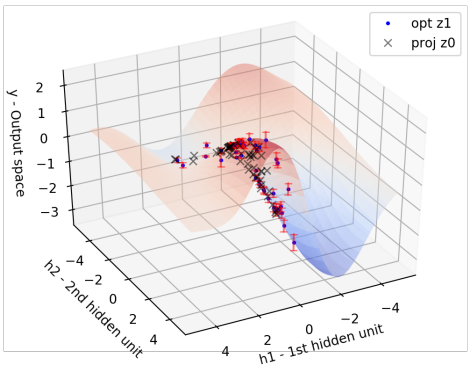
4 is no longer present as the greedy initialisation encourages an appropriate distribution of the bottom-layer inducing points $z^{(1)}$. In addition, the asymmetric greedy initialisation of the bottom layer GP-mappings guides the optimisation dynamics into a configuration where both hidden-units are asymmetric and active as shown in figures 5.5 a,b. In figure 5.5c we see how this asymmetry is combined by a non-linear top-layer GP mapping to yield the AEP-DGP predictions in figure 5.5d. Hence the greedy initialisation encourages AEP-DGPs to use their full representational capacity and avoid the tendency to use a single unit per layer described in Chapter 3.



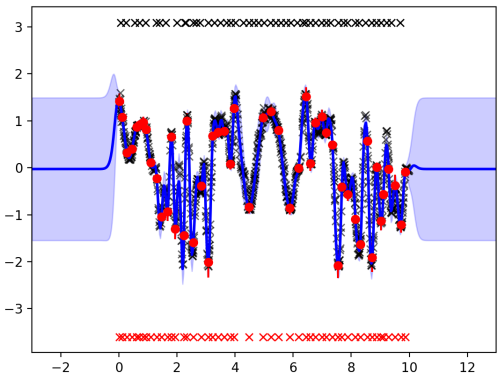
(a) $f_1^{(1)}(\mathbf{x})$ GP-mapping.



(b) $f_2^{(1)}(\mathbf{x})$ GP-mapping.



(c) $f^{(2)}(h^{(1)})$ GP-mapping.



(d) AEP-DGP regression from input to output space.

Figure 5.5 Representation of $f^{(l)}$ GP-mappings and AEP-regression using greedy initialisation.

Model		MSE	MNLPP	Sample MNLPP
Full GP	–	0.055	0.011	–
SGP	VFE	0.055	0.011	–
	FITC	0.076	3.291	–
Greedy VFE	Round 1	0.049	-0.276	0.08
	Round 2	0.049	-0.596	0.07
Greedy FITC	Round 1	0.036	-0.668	0.06
	Round 2	0.038	-0.506	0.08
Default		0.067	-0.322	0.11

Table 5.2 Reduced data domain scores with $N_{tr} = M = 50$.

5.2.1 Reduced data domains

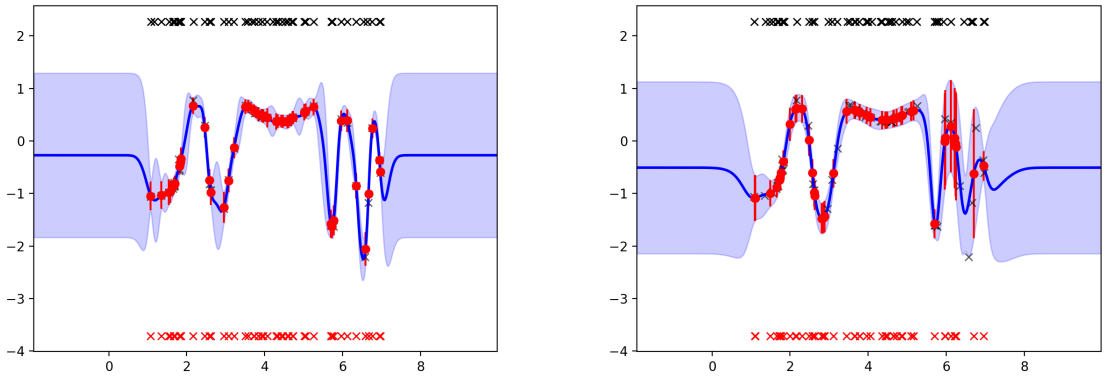
The performance of greedy AEP-DGPs depends highly on the number of inducing points and the quality of the sparse approximation used in the initialisation. We showed before in figure 5.3 that as the M increases AEP-DGP scores get closer to those of a full GP. We would be interested in knowing what happens in the limit as M approaches N_{tr}. For medium-size datasets this is computationally intractable, so we will consider a small dataset of DGP samples with $N_{tr} = 50$, while keeping a full test set of size 1000. Reduced data domains like this one are often encountered in Bayesian Optimisation, which makes them particularly interesting to study the performance of sparse DGPs and full GPs.

As discussed in Bauer et al. (2016), VFE sparse GPs recover the performance of full GPs as M approaches N_{tr}, by placing an inducing point at each training points, whereas the clumping of pseudo-points prevents this from happening in FITC sparse GPs. Table 5.2 shows the performance of greedy initialised AEP-DGPs with both types of sparse GPs compared to full GPs and the default initialisation. Both greedy initialisations outperform full GPs (with VFE and FITC providing the best MNLPP and MSE fits respectively), whereas the default initialisation does not. Figure 5.6 shows the difference between the predictions made by the default and the VFE greedy AEP-DGPs.

5.2.2 Out-of-sample test sets

Next, we study the performance of greedy AEP-DGPs in out-of-sample test sets with isolated training points, a scenario that is frequently found on high-dimensional Bayesian Optimisation problems. We remove two large flat portions of a DGP sample leaving a few points in the middle and train the model on 500 remaining points. Then, we tested our models on the out-of-sample test formed by the portions removed to produce the results on table 5.3.

We observe how despite only using 25 inducing points greedy AEP-DGPs are capable of outperforming full GPs, as long as, they locate a pseudo-point at each of the isolated points. This is



(a) Greedy initialisation using a VFE sparse GP.

(b) Default initialisation.

Figure 5.6 AEP-DGP regression on DGP sample dataset with $N_{tr} = M = 50$.

Model	MSE	MNLPP	Sample MNLPP	σ^2_{out}
Full GP	0.068	0.302	–	–
Greedy Round 2	0.042	0.125	0.015	0.013
Greedy Round 1	0.102	0.129	0.023	0.113
Bad location	0.382	1.032	0.341	0.96

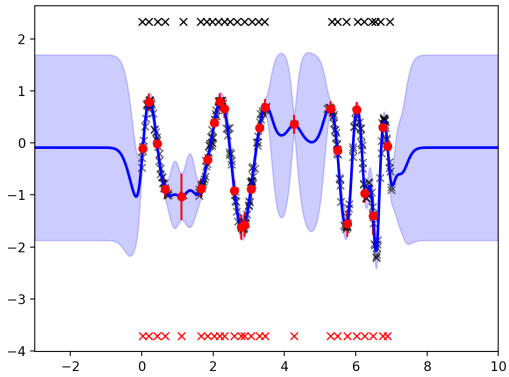
Table 5.3 Out-sample test scores for with $N_{tr} = 500$, $M = 25$.

more clearly observed in figure 5.7d where the lack of pseudo-points at isolated points causes a lot of uncertainty in that region, resulting in a poor predictive performance. Figure 5.7c (in black) shows the values of the sample based hold-out likelihood of the greedy AEP-DGP in 5.7a, which is high near training points and decreases as we distance from them.

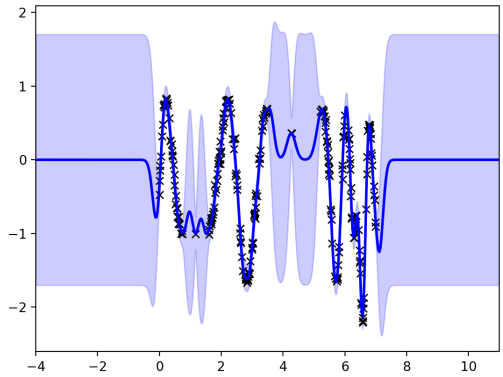
Hence we have seen that in order to ensure a tractable Expected Improvement (EI) acquisition function in Bayesian Optimisation using sparse DGPs, it is necessary to fix a pseudo-point at each isolated point. However, as the number of isolated points increases the computational cost becomes prohibited.

5.3 Conclusions

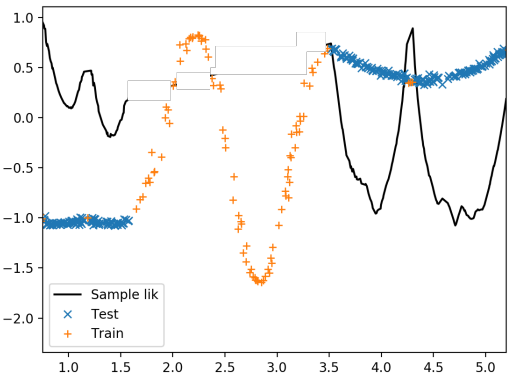
We proposed a greedy initialisation of the DGP-layers using sparse GPs, as well as, three other alternatives to the default initialisation of the `geepee` library. We tested these initialisation schemes in the same datasets as in Chapter 4, obtaining significant changes in results. Thus, we have shown the the importance of choosing a right initialisation for DGP training.



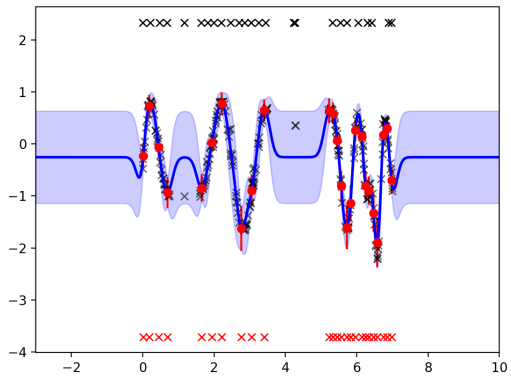
(a) AEP-DGP greedy.



(b) Full GP.



(c) Training and test sets with sample log-lik. plot.



(d) AEP-DGP missing isolated points.

Figure 5.7 Out-of-sample test set experiments.

In addition, we demonstrated how in small-size datasets our greedy initialisation is able to attain full GP performance after the first round of training and overcome full GP after the second round. These results, together with performance of greedy AEP-DGPs on out-of-sample test sets, make us believe that an implementation of sparse DGPs in Bayesian Optimisation may be plausible.

Chapter 6

Conclusions

Deep Gaussian Processes (DGPs) are powerful and flexible Bayesian non-parametric models. In order to perform inference and learning, we require approximate inference schemes that sacrifice representational capacity for tractability. An example of this is the inference scheme for sparse DGPs using approximate Expected Propagation (AEP) proposed by Bui et al. (2016) and also investigated in this thesis.

We started by giving a detailed mathematical description of the AEP-DGP model, drawing connections with related literature and understanding the limitations and properties of the approximations used. Then, we investigated the representational capacity of this model on two benchmark toy datasets, where Squared Exponential kernel Gaussian Processes (GPs) are known to have difficulties. We observed that the sparsification of the GP mappings brings complications in the optimisation process of the DGP network and can have a damaging impact on the modelling ability of AEP-DGPs.

Motivated by these pathologies, we tested the predictive performance of one-hidden-layer AEP-DGPs on medium-size datasets formed by samples generated from a DGP with the same architecture. These datasets are of special interest for our ultimate propose of implementing DGPs in Bayesian Optimisation due to the complex non-stationarity they present. Once encountered with series pathologies in the AEP-DGP predictions, we revised the training procedure, proposed alternative configurations and narrow the solution to an initialisation issue. In addition, we introduced a greedy initialisation scheme of the DGP layers that not only resolves this pathological behaviour, but also overcomes full GP performance on reduced data domains similar to those encountered in Bayesian Optimisation.

6.1 Future work

Our greedy initialisation is a first step into a potential implementation of Deep Gaussian Processes into the Bayesian Optimisation framework. This method has demonstrated exciting results in one-dimensional isolated points. Future work should question whether this initialisation scheme could be generalised to higher-dimensional settings.

One of the challenges to overcome is the special treatment of the inducing points required by this initialisation, where in the one dimensional case we place an inducing input at each of the isolated points. In higher dimensions this method may not be computationally tractable, since our greedy initialisation depends on a sparse approximation whose computational cost approaches that of a Gaussian Process as the number of isolated points increases.

Deep Gaussian Processes continue to be a very active research area in Machine Learning. With the appropriate computational resources, a Monte Carlo based inference scheme could be an alternative to approximate inference schemes like approximate Expected Propagation, for DGPs that could be efficiently implemented for Bayesian Optimisation. Perhaps a sequential inference, where the data is processed sequentially, similar to the one proposed by Wang et al. (2016) could be that alternative.

In more general terms, a question that remains to be answered is whether Deep Gaussian Process are suitable for Bayesian Optimisation. One would expect the implementation of the Expected Improvement (EI) acquisition function using DGPs (with an appropriate inference scheme) to be tractable, due to the analytical properties of DGPs. Future work should investigate whether this EI-DGPs model can compete with current state-of-the-art techniques involving Predictive Entropy Search (PES).

In the meantime, there is nothing but excitement from my part, looking forward to the world of possibilities that will come along with the development of new techniques in Bayesian Optimisation and Deep Gaussian Processes.

References

Adams, R. P. (2015). Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. *Journal of Machine Learning Research*, 37:1–6.

Alvarez, M. A., Luengo, D., Teor, D., Titsias, M. K., and Lawrence, N. D. (2011). Efficient Multioutput Gaussian Processes through Variational Inducing Kernels. *Artificial Intelligence*, (x):25–32.

Bauer, M. S., van der Wilk, M., and Rasmussen, C. E. (2016). Understanding Probabilistic Sparse Gaussian Process Approximations. *arXiv Electronic Journal*, (Nips):1–18.

Bui, T. D., Hernández-Lobato, D., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2016). Deep Gaussian Processes for Regression using Approximate Expectation Propagation. *Icml*.

Bui, T. D., Hernández-Lobato, J. M., Li, Y., Hernández-Lobato, D., and Turner, R. E. (2015). Training Deep Gaussian Processes using Stochastic Expectation Propagation and Probabilistic Backpropagation. (2):1–6.

Damianou, A. C. and Lawrence, N. D. (2013). Deep Gaussian Processes. *International Conference on Artificial Intelligence and Statistics*, 31:207–215.

Durrande, N., Ginsbourger, D., and Roustant, O. (2012). Additive covariance kernels for high-dimensional Gaussian process modeling. *Ann. Fac. Sci. Toulouse Tome 21*, 3:481–499.

Duvenaud, D., Rippel, O., Adams, R. P., and Ghahramani, Z. (2014). Avoiding Pathologies in Very Deep Networks. *Aistats*, (1):9.

Hensman, J. and Lawrence, N. D. (2014). Nested Variational Compression in Deep Gaussian Processes. pages 1–21.

Hernández-Lobato, J. M., Li, Y., Rowland, M., Hernández-Lobato, D., Bui, T., and Turner, R. E. (2015). Black-box-alpha -divergence Minimization. 48.

Li, Y., Hernandez-Lobato, J. M., and Turner, R. E. (2015). Stochastic Expectation Propagation. *Nips*, (Vi):1–14.

Minka, T. P. (2001). Expectation Propagation for Approximate Bayesian Inference. *Uncertainty in Artificial Intelligence (UAI)*, 17(2):362–369.

Quiñonero-candela, J., Rasmussen, C. E., and Herbrich, R. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1935–1959.

Rasmussen, C. E. and Williams, C. K. I. (2004). *Gaussian processes for machine learning.*, volume 14.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., Freitas, N. D., De Freitas, N., and Freitas, N. D. (2015). Taking the Human Out of the Loop : A Review of Bayesian Optimization. *RLTutor*, 104(1):1–24.

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems*, 18:1257.

Titsias, M. (2009). Variational Learning of Inducing Variables in Sparse Gaussian Processes. *Aistats*, 5:567–574.

Wang, Y., Brubaker, M., Chaib-Draa, B., and Urtasun, R. (2016). Sequential Inference for Deep Gaussian Process. *Artificial Intelligence and Statistics*, 51(2):694–703.

Warner, B. a. and Neal, R. M. (1997). Bayesian Learning for Neural Networks. *Journal of the American Statistical Association*, 92:791.