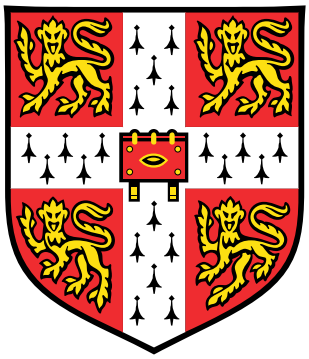
Bayesian single- and multi-objective optimisation with nonparametric priors



Amar Shah

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

St John's College June 2017

Acknowledgements

It has been a privilege to be able to undertake a doctoral degree at such a prestigious institution. Many individuals and organisations have influenced me to get to this position. Here I would like to acknowledge a few of them.

My lifelong gratitude to

My Phd supervisor, Zoubin Ghahramani, for education & inspiration,
My colleagues from CBL & worldwide, for conversation & collaboration,
My college, St John's, for accommodation & sustentation,
My many friends and family for animation & stimulation, and
My parents for building my foundation & overseeing its fortification.

Abstract

Optimisation is integral to all sorts of processes in science, economics and arguably underpins the fruition of human intelligence through millions of years of optimisation, or *evolution*. Scarce resources make it crucial to maximise their efficient usage. In this thesis, we consider the task of maximising unknown functions which we are able to query point-wise. The function is deemed to be *costly* to evaluate e.g. larger run time or financial expense, requiring a judicious querying strategy given previous observations.

We adopt a probabilistic framework for modelling the unknown function and Bayesian non-parametric modelling. In particular, we focus on the *Gaussian process* (GP), a popular non-parametric Bayesian prior on functions. We motivate these choices and give an overview of the Gaussian process in the introduction, and its application to *Bayesian optimisation*.

A GP's behaviour is intimately controlled by the choice of *kernel* or covariance function, typically chosen to be a parametric function. In chapter 2 we instead place a non-parametric Bayesian prior, known as an Inverse Wishart process prior, over a GP kernel function, and show that this may be marginalised analytically leading to a *Student-t process* (TP). Furthermore we explore a larger class of *elliptical processes*, and show that the TP is the most general for which analytic calculation is possible, and apply it successfully for Bayesian optimisation.

The remainder of the thesis focusses on various Bayesian optimisation settings. In chapter 3, we consider a setting where we are able to evaluate a function at multiple locations in parallel. Our approach is to consider a measure of information, *entropy*, to decide which batch of points to evaluate

a function at next. We similarly apply information gain for *multi-objective* Bayesian optimisation in chapter 4. Here, one wishes to find a *Pareto frontier* of efficient settings with respect to several different objectives through sequential evaluation. Finally, in chapter 5 we exploit the idea that in a multi-objective setting, functions are *correlated*, incorporating this belief in our choice of prior distribution over the multiple objectives.

Table of contents

1 Introduction **1**

1.1 Parametric Modelling 2

1.2 An Introduction to Gaussian Processes 5

1.2.1 Definition 5

1.2.2 Gaussian Processes for Regression 6

1.2.3 Gaussian Process Prediction 8

1.2.4 Hyperparameter Learning 8

1.3 An Introduction to Bayesian Optimisation 9

1.3.1 Model Choice for f 11

1.3.2 Acquisition Functions 13

1.4 Contributions 15

2 Student-t and Elliptical Processes **17**

2.1 Introduction 18

2.2 Inverse Wishart Process 20

2.3 Deriving the Student-t Process 21

2.4 Elliptical Processes 27

2.5 Deconstructing the Inverse Wishart Distribution 30

2.6 Applications 32

2.6.1 Regression 32

2.6.2 Bayesian Optimisation 37

2.7 Conclusions 41

3 Parallel Predictive Entropy Search for Batch Optimisation **43**

3.1 Introduction 44

3.1.1 Entropy 44

3.1.2 Predictive Entropy Search 45
3.2 Parallel Predictive Entropy Search (PPES) 47
3.2.1 Problem Statement and Setup 48
3.2.2 How to Approximate the Conditional Predictive Entropy 49
3.2.3 Expectation Propagation Inference 50
3.2.4 The PPES Approximation 52
3.2.5 Sampling from the Posterior over the Global Maximiser 53
3.2.6 Optimising the PPES Approximation 54
3.3 Related Work 56
3.4 Empirical Study 58
3.4.1 Assessing the Quality of the PPES Approximation . . 59
3.4.2 Comparison of Methods on Synthetic Objectives . . . 63
3.4.3 Comparison of Methods on Real World Datasets or Simulations 63
3.5 Conclusions 66

4 Predictive Entropy Search for Multi-Objective Optimisation 69
4.1 Introduction 69
4.2 Multi-objective Bayesian Optimisation via Predictive Entropy Search 72
4.3 Approximating the Conditional Predictive with Expectation Propagation 75
4.3.1 Constructing The Approximate Conditional Predictive 77
4.3.2 Computing the EP parameters 78
4.3.3 Parallel EP Updates and Damping 79
4.3.4 Sampling the Posterior Pareto Set 80
4.3.5 The Resultant Approximation 80
4.4 Related Work 81
4.5 Experiments 84
4.5.1 Accuracy of the PESMO Approximation 86
4.5.2 Experiments with Synthetic Objectives 87
4.5.3 Finding a Fast and Accurate Neural Network 91
4.5.4 Finding a Small and Accurate Ensemble of Decision Trees 94

4.6 Conclusions 99

5 Pareto Frontier Learning with Correlated Expensive Objectives 101

5.1 Pareto Hyper-volume Based Multi-objective Optimisation . . 102
5.1.1 Definition of Pareto Hyper-volume 102
5.1.2 Expected Improvement in Pareto Hyper-volume . . . 103
5.1.3 Grid Partitioning of the Output Space 105
5.2 Pareto Learning with Gaussian Processes 107
5.2.1 EIPV for Independent Gaussian Process Objectives . 107
5.2.2 Approximate EIPV for Correlated Gaussian Process Objectives 108
5.2.3 Alternative Approximation Considerations 110
5.2.4 Correlated Gaussian Process Models 111
5.3 Experiments 112
5.3.1 Empirical Illustration of the Benefit of Modelling Correlations 114
5.3.2 Quality of the CEIPV Approximation 115
5.3.3 CEIPV Performance for Varying Levels of Actual Correlation 117
5.3.4 Performance over Benchmark Multi-objective Optimisation Tasks 119
5.3.5 Performance on Real World Data and Simulations . . 120
5.4 Discussion 121

6 Conclusions 123

References 127

Chapter 1

Introduction

This thesis is about non-parameteric function modelling with Bayesian priors for the purpose of Bayesian optimisation. In this chapter, we begin by providing a high level historical overview of machine learning including research interest in neural networks and kernel machines. Next we introduce Bayesian reasoning, Gaussian process and Bayesian optimisation, before listing the subsequent contributions of this thesis in the remaining chapters.

The core of machine learning is the development of statistical and computational methods for pattern recognition of data. A practitioner may wish to use a trained model to make predictions or decisions when presented with new unseen data, which is only possible when the model has a good *inductive bias*, or generalisability.

Humans collect data about the world through continual interaction with it and develop the ability to predict the results of new actions. Therefore, techniques in machine learning are often inspired by human learning, for example, the perceptron [Rosenblatt, 1962] was inspired by a model of a human neuron [McCulloch and Pitts, 1943]. Neural networks began to be popularised in the 1980s, a key attraction being their ability to adapt basis functions, as opposed to traditional linear models with fixed basis functions [Rumelhart et al., 1986]. The expressiveness and seeming flexibility of neural network based models came at the cost of interpretability; there lacked a principled framework for making key modelling choices, which could affect

performance drastically in ways which were often difficult to explain.

In the 1990s, kernel methods grew in popularity [Cristianini and Shawe-Taylor, 2000]. Kernel methods provided a way to consider infinitely many fixed basis functions whilst utilising finite resources, thanks to the *kernel trick*, which implicitly represents inner products of basis function representations of data points using a kernel matrix [Hofmann et al., 2008].

A successful kernel based, Bayesian modelling tool for modelling non-linear functions is the *Gaussian process* [Rasmussen and Williams, 2006], which we discuss in more detail in the next subsections. In the machine learning community, Gaussian process research really spawned from the interest in neural networks. In fact, Neal [1995] argued that since model performance improved with the ability to account for additional structure in data, that we should pursue the limits of large models. He went on to show that a Gaussian process is in fact the limit of a particular Bayesian neural network architecture, as the number of hidden units tends to infinity. Gaussian process kernel machines are typically not able to scale to the size of today's large datasets. Simplifying assumptions have been studied to improve scalability [Williams and Seeger, 2001; Csató and Opper, 2002; Snelson and Ghahramani, 2006], but Gaussian processes appear to be most useful in a low to medium data setting. *Bayesian optimisation* is the task of finding the maximum of a black-box function through a small number of sequential evaluations, a task well suited to the Gaussian procees and a key focus of this thesis.

1.1 Parametric Modelling

In this section we briefly illustrate the differences between frequentist and Bayesian reasoning in a regression task, in order to subsequently motivate a Bayesian approach to function optimisation. Suppose we have a set of observation pairs $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ where y_i is the scalar observed output for a known input vector $\boldsymbol{x}_i$, and that we wish to predict the output y_* at a new input $\boldsymbol{x}_*$. The outputs may represent house prices for given inputs such as

location, house size, number of bedrooms, etc.

A typical statistical approach to learn the mapping from inputs, $\boldsymbol{x}$, to outputs, y, is to consider a class of functions which are controlled by a set of *parameters*, and learning the parameter values which lead to the best fit of the training data, for example by minimising the difference between predictions and true observed values. A simple class of functions which is able to learn linear trends is a linear model of the form $f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{x}$, where $\boldsymbol{w}$ are the parameters, or weights. For more complicated tasks, we may use a linear model with a vector of fixed non-linear basis functions, or features $\phi(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), ..., \phi_k(\boldsymbol{x})]$, with $f(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}^\top \phi(\boldsymbol{x})$. This is a linear model, since f is a linear function with respect to the parameters, but not necessarily with respect to the input space.

A common quantity to want to minimise for regression tasks where outputs are in a continuous space is

$$E(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} (f(\boldsymbol{x}_i, \boldsymbol{w}) - y_i)^2, \tag{1.1}$$

the *mean squared error* of the predictions of the model over the training data set. It is usually possible to minimise $E(\boldsymbol{x})$ using gradient descent. Often the output observations are modelled as being *noisy*, for example, a reading instrument used to measure the output of a particular process may inherently have a limit as to its accuracy due to hardware factors. For the purposes of our example, we will continue to assume noiseless observations.

A fundamental issue with the framework we have outlined above, is that for k (the number of features) very large, for example $k \gg n$, minimising (1.1) may lead to the model *overfitting*. This is where $f(\boldsymbol{x}, \boldsymbol{w})$ focuses on being able to reproduce the training data very well, caring too much on the random nuances of the training data rather than the true underlying trend, which hurts generalisation to new data. Whilst larger k leads to a more expressive regression model, it also makes overfitting more likely. A typical approach used to control the level of overfitting, is to add a complexity penalty term to the error being minimised. For example, the original objective can be

modified to

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i, \boldsymbol{w}) - y_i)^2 + \lambda \boldsymbol{w}^\top \boldsymbol{w}, \tag{1.2}$$

for a positive scalar λ. Introducing a penalty term as we have done here is called *regularisation*, the extra term acts so as to discourage weights from being too large, which in turn limits the amount of overfitting. The question becomes, "how should one set λ?" so as to trade off the error and regularisation terms. A standard way to answer this question, is to take a fraction of the training set and place it aside as a *validation set*. We would proceed to train on the remaining data with different values of λ, and validate the model by computing the error on the validation set. A search is performed over λ to minimise the error on the validation set. This process is called *cross validation*. In practice, it is unclear how one should decide on what the validation set should be, and how many different settings of λ should be validated with.

A very different approach to inference and regularisation is a *Bayesian* one, where probability distributions are used to represent subjective uncertainty of an inference algorithm [MacKay, 2003; Jaynes, 2003; Barber, 2012; Gelman et al., 2014]. Specifically, we place a *prior* distribution on the weights, $p(\boldsymbol{w})$, and now model each y_i as a sample from probability distribution e.g. $\mathrm{N}(f(\boldsymbol{x}_i, \boldsymbol{w}), \sigma^2)$ for some parameter σ. Bayes' rule tells us how to compute the posterior distribution of weights given the observed data

$$p\Big(\boldsymbol{w}|\{\boldsymbol{x}_i, y_i\}_{i=1}^n\Big) \propto p(\boldsymbol{w}) \prod_{i=1}^{n} p(y_i|\boldsymbol{w}, \boldsymbol{x}_i). \tag{1.3}$$

We subsequently may make a mean prediction at a new input point $\boldsymbol{x}_*$ by computing an integral $\mathbb{E}_{p(\boldsymbol{w}|\{\boldsymbol{x}_i, y_i\}_{i=1}^n)}[f(\boldsymbol{x}_*, \boldsymbol{w})]$, an example of Bayesian model averaging, since we are averaging the prediction over multiple function models based on how likely they are given the observed data. This averaging prevents overfitting as we are no longer considering one explanation of the data, as was the case when optimising (1.1). More generally, we may compute an entire probability distribution for our prediction at a new test point, $p\Big(y_*|\boldsymbol{x}_*, \{\boldsymbol{x}_i, y_i\}_{i=1}^n\Big)$, which enables us to consider all sorts of statistics about the prediction including the variance, mode, median, etc. Having

an entire predictive distribution over new outputs is crucial for Bayesian decision making. The stochastic nature of predictions under a Bayesian framework reflects the *uncertainty* in modelling, which at a low level may come directly from measurement errors, or at a higher level from reasoning about which model structure is ideal for the task. The probabilistic approach uses probability theory to express all forms of uncertainty. Probability theory is the mathematical language for representing and quantifying uncertainty [Ghahramani, 2015].

In this section we have discussed parametric models, which have a fixed representation capacity even as the number of datapoints increases. Non-parametric models do not have a finite number of parameters, in fact, as the number of data points increases, so do the number of parameters. The Gaussian process is a non-parametric Bayesian model which we introduce in the next section.

1.2 An Introduction to Gaussian Processes

Gaussian processes form a class of non-parametric stochastic processes over the space of functions [Rasmussen and Williams, 2006]. Unlike parametric models, the representational complexity of a Gaussian process (GP) grows with the number of observations. This is one of the many reasons which make GPs a popular choice for function modelling.

We proceed to provide a brief introduction to Gaussian process applications to statistical regression. Readers wishing to explore GPs in more detail should benefit from referring to Rasmussen and Williams [2006].

1.2.1 Definition

A Gaussian process is a collection of random variables, of which any finite number are jointly Gaussian distributed. Let $f : \mathcal{X} \to \mathbb{R}$. We write

$$f \sim \mathrm{GP}(m, k) \tag{1.4}$$

to denote that f follows a Gaussian process with *mean* function $m(\boldsymbol{x})$, and *covariance* function $k(\boldsymbol{x}, \boldsymbol{x}')$, which fully specify the Gaussian process. The function value $f(\boldsymbol{x})$ has mean $m(\boldsymbol{x})$ and has covariance with some other function value $f(\boldsymbol{x}')$ defined by $k(\boldsymbol{x}, \boldsymbol{x}')$. Therefore, for $\boldsymbol{X} \equiv [\boldsymbol{x}_1, ..., \boldsymbol{x}_n] \subset \mathcal{X}$, if we write $f_i \equiv f(\boldsymbol{x}_i)$ and $\boldsymbol{f} \equiv [f_1, ..., f_n]^\top$, we have that

$$\boldsymbol{f}|\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{m}, \mathbf{K}), \tag{1.5}$$

where $\boldsymbol{m} \equiv [m(\boldsymbol{x}_1), ..., m(\boldsymbol{x}_n)]^\top$ and $\mathbf{K}$ is a matrix with (i, j) entry $K_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

1.2.2 Gaussian Processes for Regression

Regression problems are core to statistical inference and machine learning. The goal is to learn a mapping from inputs to outputs based on a set of observed input-output pairs $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$. A typical application of learning this mapping is to subsequently be able to predict an output y_* at a new test input $\boldsymbol{x}_*$. We describe how one may use a Gaussian process and Bayesian inference to respectively model and learn the function mapping from inputs to outputs.

As we described previously, Bayesian inference consists of placing a prior distribution on parameters of interest, observing data and finally computing the posterior belief about parameters given the data. In regression tasks, the goal is to infer the function which maps inputs to outputs. A parametric approach to Bayesian regression is to consider a family of functions parametrised by a finite set of random variables over which inference is performed. A less confined approach to Bayesian inference would be to directly specify a prior over an infinite-dimensional space of functions, instead of implicitly specifying a prior over functions via parameters [Orbanz and Teh, 2011].

A Gaussian process is a very popular choice as a prior over continuous functions. Suppose $f \sim \mathrm{GP}(m, k)$ and that we evaluate the function at points $\boldsymbol{x}_1, ..., \boldsymbol{x}_n \in \mathcal{X}$, then $\boldsymbol{f}|\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{m}, \mathbf{K})$ as described above. Further suppose that each input location $\boldsymbol{x}_i$, we observe a real value y_i which is

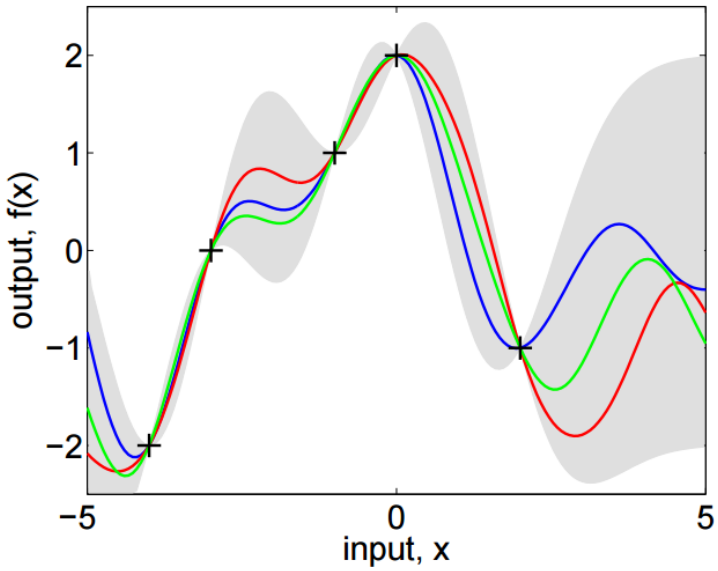


Fig. 1.1 Gaussian process posterior from 5 noiseless observations (plus signs). Coloured lines represent some possible functions to explain the observed data. The shaded region represents a 95 % predictive interval of where the function lies given observations. Figure from Rasmussen and Williams [2006].

stochastically dependent on the underlying function value $f(\boldsymbol{x}_i)$. Bayes' theorem informs us on how to obtain a posterior distribution of the function values at locations $\boldsymbol{x}_i$ given the observations y_i:

$$p(\boldsymbol{f}|\boldsymbol{y},\boldsymbol{X}) = \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{X})}{p(\boldsymbol{y}|\boldsymbol{X})} = \frac{p(\boldsymbol{y}|\boldsymbol{f})\mathcal{N}(\boldsymbol{f};\boldsymbol{m},\mathbf{K})}{p(\boldsymbol{y}|\boldsymbol{X})}. \tag{1.6}$$

A typical scenario is that the observations y_i are simply the true function values corrupted by Gaussian noise, i.e. $y_i|f_i \sim \mathcal{N}(f_i,\sigma^2)$. In this case, since both the likelihood $p(y_i|f_i)$ and the prior $p(\boldsymbol{f})$ are Gaussian, we can analytically compute the posterior distribution

$$\boldsymbol{f}|\boldsymbol{y},\boldsymbol{X} \sim \mathcal{N}\Big(\boldsymbol{m} + \mathbf{K}(\mathbf{K}+\sigma^2\boldsymbol{I})^{-1}(\boldsymbol{y}-\boldsymbol{m}), \mathbf{K} - \mathbf{K}(\mathbf{K}+\sigma^2\boldsymbol{I})^{-1}\mathbf{K}\Big), \tag{1.7}$$

which is in fact multivariate Gaussian distributed. Figure 1.1, taken from Rasmussen and Williams [2006], shows an example of some possible functions which could explain 5 noiseless observations, along with a 95 % predictive interval.

1.2.3 Gaussian Process Prediction

Whilst the posterior distribution of the function f at observed input locations $\boldsymbol{X}$ is useful, what is often of more interest is the predictive distribution at new input locations. Let $\boldsymbol{X}^* \equiv \{\boldsymbol{x}_1^*, ..., \boldsymbol{x}_{n'}^*\} \subset \mathcal{X}$ be a set of new input locations. Let $f_i^* \equiv f(\boldsymbol{x}_i^*)$ and $\boldsymbol{f}^* \equiv [f_1^*, ..., f_{n'}^*]^\top$. Further let the concatenation of old and new input locations be $\boldsymbol{X}_+ \equiv \boldsymbol{X} \cup \boldsymbol{X}^*$, and let $\boldsymbol{f}_+ \equiv [f_1, ..., f_n, f_1^*, ..., f_{n'}^*]^\top$. We write $\boldsymbol{m}^* \equiv [m(\boldsymbol{x}_1^*), ..., m(\boldsymbol{x}_{n'}^*)]^\top$ and $\boldsymbol{m}_+$ as the concatenation of $\boldsymbol{m}$ and $\boldsymbol{m}^*$. Further let $\mathbf{K}_+$ be a $(n + n') \times (n + n')$ covariance matrix such that

$$\mathbf{K}_+ = \begin{pmatrix} \mathbf{K} & \mathbf{K}' \\ \mathbf{K}'^\top & \mathbf{K}^* \end{pmatrix}, \tag{1.8}$$

where $K_{i,j}^* = k(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*)$ for $i, j \in \{1, ..., n'\}$, and $K'_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j^*)$ for $i \in \{1, ..., n\}$ and $j \in \{1, ..., n'\}$. Then

$$p(\boldsymbol{f}^*|\boldsymbol{y}, \boldsymbol{X}_+) = \int p(\boldsymbol{f}^*|\boldsymbol{f}, \boldsymbol{X}_+) p(\boldsymbol{f}|\boldsymbol{y}, \boldsymbol{X}) d\boldsymbol{f}, \tag{1.9}$$

and

$$\boldsymbol{f}^*|\boldsymbol{y}, \boldsymbol{X}_+ \sim \mathcal{N}\Big(\mathbf{K}'^\top (\mathbf{K} + \sigma^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \boldsymbol{m}) + \boldsymbol{m}^*, \tag{1.10}$$

$$\mathbf{K}^* - \mathbf{K}'^\top (\mathbf{K} + \sigma^2 \boldsymbol{I})^{-1} \mathbf{K}'\Big). \tag{1.11}$$

Thus the posterior predictive distribution of a Gaussian process conditioned on observations is itself a Gaussian process.

1.2.4 Hyperparameter Learning

The behaviour of a draw from a Gaussian process prior is predominantly encoded in the choice of the covariance function, k. Typically, the Gaussian process mean is chosen to be zero (or constant), and the covariance function is chosen to be parametric, such that $k = k_\theta$, for some set of parameters θ. The parameters of the covariance function are often referred to as hyperparameters of the Gaussian process model. Hyperparameters control the nature of the covariance between points in $\mathcal{X}$, for example, length-scale hyperparameters control how quickly covariance decays as a function of

$|x_d - x'_d|$. The hyperparameters can be learnt by maximising likelihood, i.e. by optimising

$$p(\boldsymbol{y}|\boldsymbol{X}, \theta) = -\frac{1}{2}\boldsymbol{y}^\top(\mathbf{K}_\theta + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y} - \frac{1}{2}\log|\mathbf{K}_\theta + \sigma^2\boldsymbol{I}| - \frac{n}{2}\log 2\pi \quad (1.12)$$

with respect to θ by gradient ascent. Alternatively, one may apply a fully Bayesian treatment to the hyperparameters by imposing a prior distribution on them, and sampling from the posterior

$$p(\theta|\boldsymbol{y}, \boldsymbol{X}) \propto p(\boldsymbol{y}|\boldsymbol{X}, \theta)p(\theta). \quad (1.13)$$

1.3 An Introduction to Bayesian Optimisation

Many tasks in engineering and science can be abstractly described in terms of optimising a nonlinear function $f(\boldsymbol{x})$ over a compact set $\mathcal{X} \subset \mathbb{R}^d$. More precisely, the task can be described as

$$\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}). \quad (1.14)$$

In many instances, f is known in analytically and may also be differentiable, in which case gradient based optimisation is often employed to find the maximum of the function. However, there exist many scenarios where f is not known analytically, and may only be evaluated pointwise through a simulation or process requiring many resources e.g. time or money.

In a scenario where pointwise evaluation is costly, a strategy of evaluating at points in $\mathcal{X}$ uniformly at random is naive and wasteful, and would converge to large values of f far too slowly as the dimensionality of the input space increases. Let $\boldsymbol{x}^+$ denote the best observed input location through sequential evaluation, and let $\boldsymbol{x}^*$ denote the true maximiser of the function f. In fact, Betrò [1991] showed that an objective function with Lipschitz continuity C would require $\mathcal{O}\Big((C/2\epsilon)^d\Big)$ random uniform evaluations to ensure that $f(\boldsymbol{x}^+) \geq f(\boldsymbol{x}^*) - \epsilon$.

Algorithm 1 Bayesian Optimisation

1: $\mathcal{D} \leftarrow \emptyset$
2: **for** $t = 1, 2, ..., T$ **do**
3: $\quad \boldsymbol{x}_t \leftarrow \text{argmax}_{\boldsymbol{x} \in \mathcal{X}}\, u(\boldsymbol{x}|\mathcal{D})$
4: $\quad f_t \leftarrow f(\boldsymbol{x}_t)$
5: $\quad \mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{x}_t, f_t\}$
6: $\boldsymbol{x}^+ \leftarrow \text{argmax}_{\boldsymbol{x} \in\, \boldsymbol{x}_{1:T}} f(\boldsymbol{x})$
7: **return** $\boldsymbol{x}^+$

A Bayesian approach to optimising f, would be to model it by placing a prior distribution over it, which would be updated sequentially using Bayes' rule as function evaluations are observed. The posterior model of f given function evaluations is subsequently used to decide where in the input domain, $\mathcal{X}$, to evaluate the function next, in order to maximise a chosen analytically tractable criterion which trades off exploration and exploitation. The criterion, $u(\boldsymbol{x})$, is often referred to as the acquisition function. A basic version of the Bayesian optimisation procedure is outlined in Algorithm 1 and illustrated in Figure 1.2 taken from [Brochu et al., 2009], which offers a nice and more thorough introduction to Bayesian optimisation.

Bayesian optimisation research has been fuelled by many successful applications in a diverse range of tasks. A few such successes are listed below:

- **Robotics and reinforcement learning.** Parametrising a robot's gait parameters allows you to optimise over its velocity or smoothness using Bayesian optimisation [Lizotte, 2008]. Similarly, robotic policy parametrisation and search techniques can be used to navigate a robot through landmarks whilst minimising uncertainty about location and map estimation [Martinez-Cantin et al., 2007].

- **Natural language, processing and text.** Bayesian optimisation has been used to improve text extraction [Wang et al., 2014] and to tune representations for general text and language tasks [Yogatama and Smith, 2015].

- **Sensor placement.** Meteorological sensors can be expensive to obtain, so intelligent placement of them for the task at hand is useful.

Bayesian optimisation has been used to tackle this problem [Garnett et al., 2010]. For mobile sensing, a cost can be associated in making a measurement (e.g. on a unmanned autonomous vehicle), this cost can be incorporated in decision making processes using Bayesian optimisation [Marchant and Ramos, 2012].

- **Automatic machine learning.** The goal is to automatically search over hyperparameters of machine learning algorithms for a given task. For example, when training a neural network, one has to choose a learning rate, momentum, weight decay, etc. Cross validation can be very expensive when a lot of parameters have to be searched over. Bayesian optimisation has recently been successfully applied to search over the space of hyperparameters, in some cases finding state of the art settings [Snoek et al., 2012].

The two key design choices required for implementing a Bayesian optimisation algorithm are: (i) the model for f, and (ii) the *acquisition function* to decide where to evaluate the function next. We discuss each of these choices in the following subsections.

1.3.1 Model Choice for f

Gaussian processes have been the most popular choice of prior over the function f as they are non-parametric and permit analytically tractable computation, as we will see in the next subsection. The use of Gaussian process priors for Bayesian optimisation began in the late 1970s [O'Hagan, 1978; Zilinskas, 1980].

The choice of covariance function is integral to the Gaussian process prior as it determines the smoothness properties of drawn samples. Various forms of covariance functions are discussed in Rasmussen and Williams [2006]. For isotropic function modelling, the squared exponential kernel with single hyperparameter θ is most common, and defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\Big(-\frac{1}{2\theta^2}||\boldsymbol{x} - \boldsymbol{x}'||^2\Big). \tag{1.15}$$

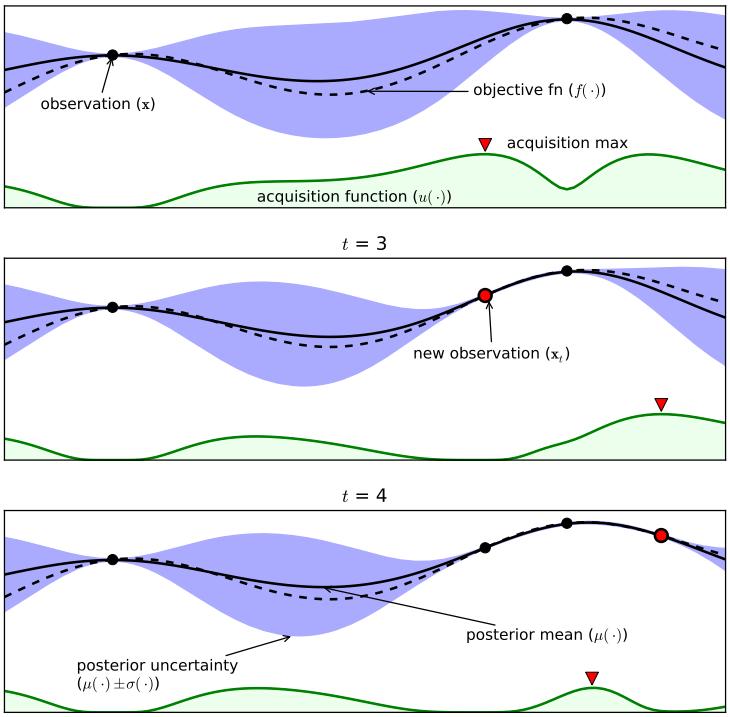


Fig. 1.2 An example of Gaussian process based Bayesian optimisation on a 1D objective to be maximised. Figures show 3 iterations of sampled values from the objective function from top to bottom. Observations are represented by dots, the true objective is the dotted line, the GP posterior mean and 95 % predictive interval are given by the dark solid lines and the purple shaded regions respectively. The green line at the bottom of each plot shows the acquisition function at each iteration. The acquisition is high where the GP predicts a high function value (exploitation) and where the predictive uncertainty is high (exploration). Figure from Brochu et al. [2009].

Most applications require anisotropic models which do not assume differences in each component of $\boldsymbol{x}$ contribute to the covariance equally. In such a scenario, the squared exponential with automatic relevance determination (ARD) hyperparameters $\boldsymbol{\theta}$ is a popular choice, defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^\top \mathrm{diag}(\theta)^{-2}(\boldsymbol{x} - \boldsymbol{x}')\right), \tag{1.16}$$

where $\mathrm{diag}(\theta)$ is a diagonal matrix with d^{th} diagonal entry θ_d. We can see intuitively that if a particular θ_d has a small value, the covariance becomes

less dependant on variations in the d^{th} dimension of $\boldsymbol{x}$, hence the kernel function determines the relevance of each input dimension. One drawback of the squared exponential kernels is that they lead to very smooth functions which are infinitely differentiable. The Matérn kernel [Matérn, 1986] attempts to alleviate this problem by incorporating a smoothness parameter ζ, and is defined as

$$k(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{2^{\zeta-1}\Gamma(\zeta)}\Big(2\sqrt{\zeta}||\boldsymbol{x} - \boldsymbol{x}'||\Big)^{\zeta} H_{\zeta}\Big(2\sqrt{\zeta}||\boldsymbol{x} - \boldsymbol{x}'||\Big) \tag{1.17}$$

where $\Gamma(.)$ and $H_{\zeta}(.)$ are the Gamma function and the Bessel function of order ζ. As $\zeta \to \infty$, the Matérn kernel reduces to the squared exponential kernel. It is common to incorporate ARD hyperparameters into the Matérn kernel in practice. Kernels can be formed by composing other kernels.

1.3.2 Acquisition Functions

The *acquisition function* controls the search for the optimum of the unknown objective function given the model. Typically, an acquisition function $u(\boldsymbol{x}|\mathcal{D})$ defines the current expected utility of evaluating f at $\boldsymbol{x}$ given the observed data so far, $\mathcal{D}$, is defined such that values of high acquisition correspond to input locations which offer more utility in evaluating the function at next. Hence, maximising the acquisition function tells one where to evaluate f next i.e. $\boldsymbol{x}_t = \text{argmax}_{\boldsymbol{x}} u(\boldsymbol{x}|\mathcal{D})$.

In this subsection, we will describe 3 of the most common choices of acquisition functions for Bayesian optimisation tasks. We assume a Gaussian process prior over the function f, and that we have observed the pairs $\mathcal{D} = \{\boldsymbol{x}_s, f_s\}_{s=1}^{T}$, with $\boldsymbol{x}^+ = \text{argmax}_{\boldsymbol{x} \in \boldsymbol{x}_{1:T}} f(\boldsymbol{x})$. We denote the posterior mean and covariance of f given observations $\mathcal{D}$ at location $\boldsymbol{x}$ as $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})^2$.

Kushner [1964] suggested maximising the probability of improving over the current best evaluation $f(\boldsymbol{x}^+)$ with some margin, $\tau \geq 0$, defining an

acquisition function

$$\begin{aligned} u_{\mathrm{PI}}(\boldsymbol{x}|\mathcal{D}) &= \mathbb{P}(f(\boldsymbol{x}) \geq f(\boldsymbol{x}^+) + \tau) \\ &= \Phi\left(\frac{\mu(\boldsymbol{x}) - f(\boldsymbol{x}^+) - \tau}{\sigma(\boldsymbol{x})}\right) \end{aligned} \tag{1.18}$$

where $\Phi(.)$ is the standard normal cumulative distribution function. Large values of τ lead to more exploration whilst small values encourage exploitation, leading Kushner to recommend a schedule for τ such that it starts out fairly high in the optimisation procedure and is decreased to zero as the algorithm progresses. Note that the probability of improvement acquisition function is indeed differentiable with respect to $\boldsymbol{x}$, making gradient ascent of the acquisition function possible.

Jones [2001b] studied the impact of the τ parameter on the performance of Bayesian optimisation algorithm and concluded that it was highly sensitive. A more satisfying approach would weigh up the magnitude of improvement with the probability of improvement. In this vein, Mockus et al. [1978] defined the expected improvement over the current best evaluation as

$$\begin{aligned} u_{\mathrm{EI}}(\boldsymbol{x}|\mathcal{D}) &= \mathbb{E}\Big[\max\Big(f(\boldsymbol{x}) - f(\boldsymbol{x}^+), 0\Big)\Big] \\ &= \sigma(\boldsymbol{x})\Big[\kappa(\boldsymbol{x})\Phi(\kappa(\boldsymbol{x})) + \phi(\kappa(\boldsymbol{x}))\Big] \end{aligned} \tag{1.19}$$

where $\kappa(\boldsymbol{x}) = \Big(\mu(\boldsymbol{x}) - f(\boldsymbol{x}^+)\Big)/\sigma(\boldsymbol{x})$ and $\phi(.)$ is the standard normal density function. The expected improvement acquisition function elegantly weighs up both the probability and magnitude of improvement over the current best. Another fairly simple way to trade off exploration and exploitation is to maximise an upper confidence bound of the form

$$u_{\mathrm{UCB}}(\boldsymbol{x}|\mathcal{D}) = \mu(\boldsymbol{x}) + \beta_t \sigma(\boldsymbol{x}) \tag{1.20}$$

where β_t is $\mathcal{O}\Big(\sqrt{d \log t}\Big)$ [Srinivas et al., 2010]. Note that each of the acquisition functions presented here are *myopic*, in the sense that they maximise a quantity based on a one-step ahead reward. The task in Bayesian optimisation is to find an optimiser within T steps, or more concretely, to minimise

$f(\boldsymbol{x}^*) - f(\boldsymbol{x}^+)$ after T steps. However, sampling multiple steps ahead quickly becomes computationally infeasible due to the combinatorial explosion of possibilities over multiple steps. Nevertheless, Ginsbourger et al. [2008] did analytically compute expected improvement for the two-step ahead problem.

Bayesian optimisation is intimately related to the problem of multi-armed bandits, where at each time step, a decision maker chooses an action from an action set, $a_t \in \mathcal{A}$, and receives a reward r_t from the environment. The actions are commonly described as arms in an analogy to the choice a gambler makes when deciding between several slot machines (one armed-bandits), which has a random payoff dependent on the particular arm. The gambler chooses so as to maximise their long term reward, by exploiting arms which pay well and exploring arms for which we require more information [Robbins, 1952; Gittins, 1979]. There have been many successful applications of multi-armed bandits including clinical trials [Thompson, 1933] and scheduling [Veatch and Wein, 1996].

1.4 Contributions

Here I outline the key contributions in this dissertation. Most of the work has been published in peer-reviewed conferences, but additional details and derivations are included in this document.

Previously in this chapter, we introduced the Gaussian process. However, the Gaussian distribution is only one of a large class of *elliptical distributions*. In chapter 2, we explore the possibility of more general elliptical processes, and prove that the Student-t process is the most general elliptical process which permits analytically tractable predictions. We subsequently show that the Student-t process is useful for applications with heavy tailed noise and also for Bayesian optimisation.

In chapter 3 we develop a novel acquisition function for Bayesian optimisation where we are able to evaluate an objective function at multiple points in parallel. The acquisition function uses the notion of information

gain and statistical entropy in deciding at which points to evaluate next. Our method is the only one known to us which is able to choose a batch of points together, rather than one by one in a greedy fashion.

A similar but modified entropy based approach can be used in the multi-objective optimisation setting. In chapter 4, we discuss the notion of a Pareto frontier of optimal points when jointly trying to optimise multiple objectives, and show how one can choose to evaluate multiple objectives at a location which maximises the information gain about the true Pareto frontier.

Finally in chapter 5, we model correlations between objective functions for the purpose of multi-objective optimisation. This proves to be fruitful in realistic scenarios as the multiple objective one may wish to optimise jointly are typically inherently correlated. In this work, we consider a volume based generalisation of the expected improvement acquisition function, and derive an analytically tractable approximation to the desired acquisition function.

Chapter 2

Student-t and Elliptical Processes

In the introduction, I discuss the Gaussian process and the parameters which control its behaviour. A key design choice in prescribing a Gaussian process prior is in the covariance function. Whilst the kernel function is almost always chosen to be a parametric function, in this chapter we consider a non-parametric, stochastic alternative called the inverse Wishart process (IWP), which attempts to *learn* the most appropriate form of covariance function based on the training data. Marginalising the IWP prior on the Gaussian process covariance function leads to a *Student-t process*.

A connection to a more general class called *elliptical distributions* is made, and we construct a Student-t process via an alternative generative process, shedding light on some of the redundancies of the inverse Wishart process framework. The GP with IWP covariance function is designed to be able to flexibly model the covariance between function points and therefore is ideally suited to Bayesian optimisation, which is our primary application area.

The core of this chapter was research I conducted in collaboration with Andrew Gordon Wilson and Zoubin Ghahramani, and has been published at AISTATS [Shah et al., 2014]. The majority of the work including the parameterisation of the IWP, the experiments and the geometric interpretation

of the IWP were developed by myself, whilst Andrew and Zoubin offered guiding thoughts and suggestions for applications.

2.1 Introduction

Gaussian processes are rich distributions over functions, which provide a Bayesian non-parametric approach to regression. Owing to their interpretability, non-parametric flexibility, large support, consistency, simple exact learning and inference procedures, and impressive empirical performances [Rasmussen, 1996], Gaussian processes as kernel machines have steadily grown in popularity over the last decade.

At the heart of every Gaussian process (GP) is a parametrised covariance kernel, which determines the properties of likely functions under a GP. Typically simple parametric kernels, such as the Gaussian (squared exponential) kernel are used, and its parameters are determined through marginal likelihood maximisation, having analytically integrated away the Gaussian process. However, a fully Bayesian non-parametric treatment of regression would place a non-parametric prior over the Gaussian process covariance kernel, to represent uncertainty over the kernel function, and to reflect the natural intuition that the kernel does not have a simple parametric form.

Likewise, given the success of Gaussian processes kernel machines, it is also natural to consider more general families of elliptical processes [Fang et al., 1989], such as Student-t processes, where any collection of function values has a desired elliptical distribution, with a covariance matrix constructed using a kernel.

As we will show, the Student-t process can be derived by placing an inverse Wishart process prior on the kernel of a Gaussian process. Given their intuitive value, it is not surprising that various forms of Student-t processes have been used in different applications [Yu et al., 2007; Zhang and Yeung, 2010; Xu et al., 2011; Archambeau and Bach, 2010]. However, the connec-

tions between these models and their theoretical properties, remain largely unknown. Similarly, the practical utility of such models remains uncertain. For example, Rasmussen and Williams [2006] wonder whether "the Student-t process is perhaps not as exciting as one might have hoped".

In this work, we precisely define and motivate the inverse Wishart process [Dawid, 1981] as a prior over covariance matrices of arbitrary size, and derive a Student-t process (TP), from hierarchical Gaussian process models. Our Student-t process has analytic forms of marginal and predictive distributions, and analytic derivatives of the marginal likelihood. We go on to define elliptical distributions and corresponding processes, and prove that the Student-t process is the most general elliptically symmetric process with analytic marginal and predictive distributions. This result allowed us to derive a new way of sampling from the inverse Wishart process, which intuitively resolves the seemingly bizarre marginal equivalence between inverse Wishart and inverse Gamma priors for covariance kernels in hierarchical GP models. Unlike for a GP, we show that the predictive covariance of a TP depend on the values of training observations. Contrary to the Student-t process described in Rasmussen and Williams [2006], we are able to derive an analytic TP noise model which can be used to separate signal and noise analytically.

The TP appears to be more robust to change-points and model misspecification than the GP, and seems to have notably improved predictive covariance. The TP further supports useful *tail-dependence* between distant function values (which is separate from the choice of kernel). Both of these properties make the TP particularly promising for Bayesian optimisation, where predictive covariance are especially important, as we show in Section 2.6.2.

We proceed by introducing and defining the inverse Wishart process, and then apply it as a prior over a GP covariance function, leading to the Student-t process. Next we discuss elliptical distributions and processes, drawing on some interesting connections to our initial approach to deriving a Student-t process. Finally we discuss some of our experimental findings and discuss where a Student-t process appears most useful.

2.2 Inverse Wishart Process

We claim that the inverse Wishart distribution is an attractive choice of prior for covariance matrices of arbitrary size. The Wishart distribution is a probability distribution over $\Pi(n)$, the set of real valued, $n \times n$, symmetric, positive definite matrices. Its density function is defined as follows.

Definition 1. *A random $\mathbf{\Sigma} \in \Pi(n)$ is Wishart distributed with parameters $\nu > n-1$, $\mathbf{K} \in \Pi(n)$, and we write $\mathbf{\Sigma} \sim \mathrm{W}_n(\nu, \mathbf{K})$ if its density is given by*

$$p(\mathbf{\Sigma}) = c_n(\nu, \mathbf{K})|\mathbf{\Sigma}|^{(\nu-n-1)/2} \exp\Big(-\frac{1}{2}\mathrm{Tr}\Big(\mathbf{K}^{-1}\mathbf{\Sigma}\Big)\Big), \tag{2.1}$$

where $c_n(\nu, \mathbf{K}) = \Big(|\mathbf{K}|^{\nu/2}2^{\nu n/2}\Gamma_n(\nu/2)\Big)^{-1}$.

The Wishart distribution defined with this parametrisation is consistent under marginalisation. If $\mathbf{\Sigma} \sim \mathrm{W}_n(\nu, K)$, then any $n_1 \times n_1$ principal submatrix, $\mathbf{\Sigma}_{11}$, is $\mathrm{W}_{n_1}(\nu, K_{11})$ distributed. This property is one which makes the Wishart distribution appear to be an attractive prior over covariance matrices. Unfortunately the Wishart distribution suffers a flaw which makes it impractical for non-parametric Bayesian modelling.

Suppose we wish to model a covariance matrix using $\nu^{-1}\mathbf{\Sigma}$, with expected value $\mathbb{E}[\nu^{-1}\mathbf{\Sigma}] = \mathbf{K}$, and $\mathrm{var}[\nu^{-1}\mathbf{\Sigma}_{ij}] = \nu^{-1}(\mathbf{K}_{ij}^2 + \mathbf{K}_{ii}\mathbf{K}_{jj})$. Since we require $\nu > n-1$, we must let $\nu \to \infty$ to define a process which has positive semi-definite Wishart distributed marginals of arbitrary size. However, as $\nu \to \infty$, $\nu^{-1}\mathbf{\Sigma}$ tends to the constant matrix $\mathbf{K}$ almost surely. Thus the requirement $\nu > n-1$ prohibits defining a useful process which has Wishart marginals of arbitrary size. Nevertheless, the *inverse Wishart* distribution does not suffer this problem. Dawid [1981] parametrised the inverse Wishart distribution as follows:

Definition 2. *A random $\mathbf{\Sigma} \in \Pi(n)$ is inverse Wishart distributed with parameters $\nu \in \mathbb{R}_+$, $\mathbf{K} \in \Pi(n)$ and we write $\mathbf{\Sigma} \sim \mathrm{IW}_n(\nu, \mathbf{K})$ if its density is given by*

$$p(\mathbf{\Sigma}) = c_n(\nu, \mathbf{K})|\mathbf{\Sigma}|^{-(\nu+2n)/2} \exp\Big(-\frac{1}{2}\mathrm{Tr}\Big(K\mathbf{\Sigma}^{-1}\Big)\Big), \tag{2.2}$$

with $c_n(\nu, K) = \dfrac{|K|^{(\nu+n-1)/2}}{2^{(\nu+n-1)n/2}\Gamma_n((\nu+n-1)/2)}$.

If $\mathbf{\Sigma} \sim \mathrm{IW}_n(\nu, \mathbf{K})$, $\mathbf{\Sigma}$ has mean and covariance only when $\nu > 2$ and $\mathbb{E}[\mathbf{\Sigma}] = (\nu-2)^{-1}\mathbf{K}$. Both the Wishart and the inverse Wishart distributions place prior mass on every $\mathbf{\Sigma} \in \Pi(n)$. Furthermore $\mathbf{\Sigma} \sim \mathrm{W}_n(\nu, \mathbf{K})$ if and only if $\mathbf{\Sigma}^{-1} \sim \mathrm{IW}_n(\nu - n + 1, \mathbf{K}^{-1})$.

Dawid [1981] shows that the inverse Wishart distribution defined as above is consistent under marginalisation. If $\mathbf{\Sigma} \sim \mathrm{IW}_n(\nu, \mathbf{K})$, then any principal submatrix $\mathbf{\Sigma}_{11}$ will be $\mathrm{IW}_{n_1}(\nu, \mathbf{K}_{11})$ distributed. Note the key difference in the parametrisation of both distributions: the parameter ν does not need to depend on the size of the matrix in the inverse Wishart distribution. These properties are desirable and motivate defining a process which has inverse Wishart marginals of arbitrary size. Let $\mathcal{X}$ be some input space and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ a positive definite kernel function.

Definition 3. $\mathbf{\Sigma}$ *is an inverse Wishart process on* $\mathcal{X}$ *with parameters* $\nu \in \mathbb{R}_+$ *and base kernel* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *if for any finite collection* $x_1, ..., x_n \in \mathcal{X}$, $\mathbf{\Sigma}(x_1, ..., x_n) \sim \mathrm{IW}_n(\nu, \mathbf{K})$ *where* $\mathbf{K} \in \Pi(n)$ *with* $\mathbf{K}_{ij} = k(x_i, x_j)$. *We write* $\mathbf{\Sigma} \sim \mathrm{IWP}(\nu, k)$.

In the next section we use the inverse Wishart process as a non-parametric prior over kernels in a hierarchical Gaussian process model.

2.3 Deriving the Student-t Process

Gaussian processes (GPs) are characterised by a mean function and a kernel function. Practitioners tend to use parametric kernel functions and learn their hyperparameters using maximum likelihood or sampling based methods. We propose placing an inverse Wishart process prior on the kernel function, leading to a Student-t process.

For a base kernel function k_θ parametrised by θ, and a continuous mean function $\phi : \mathcal{X} \to \mathbb{R}$, our generative approach is as follows

$$\begin{aligned}\sigma &\sim \mathrm{IWP}(\nu, k_\theta)\\ y|\sigma &\sim \mathrm{GP}(\phi, (\nu-2)\sigma),\end{aligned} \tag{2.3}$$

where σ is an infinite dimensional covariance function defined on $\mathcal{X} \times \mathcal{X}$. Since the inverse Wishart distribution is a conjugate prior for the covariance matrix of a Gaussian likelihood, we can analytically marginalise σ in the generative model of (2.3). For any collection of data $\boldsymbol{y} = (y_1, ..., y_n)^\top$ with $\boldsymbol{\phi} = (\phi(x_1), ..., \phi(x_n))^\top$,

$$\begin{aligned}p(\boldsymbol{y}|\nu, \mathbf{K}) &= \int p(\boldsymbol{y}|\boldsymbol{\Sigma}) p(\boldsymbol{\Sigma}|\nu, \mathbf{K}) d\boldsymbol{\Sigma}\\ &\propto \int \frac{\exp\left(-\frac{1}{2}\mathrm{Tr}\left(\left(\mathbf{K} + \frac{1}{\nu-2}(\boldsymbol{y}-\boldsymbol{\phi})(\boldsymbol{y}-\boldsymbol{\phi})^\top\right)\boldsymbol{\Sigma}^{-1}\right)\right)}{|\boldsymbol{\Sigma}|^{(\nu+2n+1)/2}} d\boldsymbol{\Sigma}\\ &\propto \left(1 + \frac{1}{\nu-2}(\boldsymbol{y}-\boldsymbol{\phi})^\top \mathbf{K}^{-1}(\boldsymbol{y}-\boldsymbol{\phi})\right)^{-(\nu+n)/2}.\end{aligned} \tag{2.4}$$

We leverage this result to define a Student-t process as follows.

Definition 4. *$\boldsymbol{y} \in \mathbb{R}^n$ is multivariate Student-t distributed with parameters $\nu \in \mathbb{R}_+ \backslash [0, 2]$, $\boldsymbol{\phi} \in \mathbb{R}^n$ and $\mathbf{K} \in \Pi(n)$ if it has density*

$$p(\boldsymbol{y}) = \frac{\Gamma(\frac{\nu+n}{2})}{((\nu-2)\pi)^{\frac{n}{2}}\Gamma(\frac{\nu}{2})}|\mathbf{K}|^{-1/2} \times \left(1 + \frac{(\boldsymbol{y}-\boldsymbol{\phi})^\top \mathbf{K}^{-1}(\boldsymbol{y}-\boldsymbol{\phi})}{\nu-2}\right)^{-\frac{\nu+n}{2}} \tag{2.5}$$

We write $\boldsymbol{y} \sim \mathrm{MVT}_n(\nu, \boldsymbol{\phi}, \mathbf{K})$.

The mean and covariance of the MVT are easily computed using the generative derivation:

$$\begin{aligned}\mathbb{E}[\boldsymbol{y}] &= \mathbb{E}[\mathbb{E}[\boldsymbol{y}|\boldsymbol{\Sigma}]] = \boldsymbol{\phi}\\ \mathrm{cov}[\boldsymbol{y}] &= \mathbb{E}[\mathbb{E}[(\boldsymbol{y}-\boldsymbol{\phi})(\boldsymbol{y}-\boldsymbol{\phi})^\top|\boldsymbol{\Sigma}]] = \mathbb{E}[(\nu-2)\boldsymbol{\Sigma}] = \mathbf{K}.\end{aligned} \tag{2.6}$$

It is also straightforward to show that the Student-t is consistent under marginalisation, which we do in the following Lemma.

Lemma 5. *The multivariate Student-t is consistent under marginalisation.*

Proof. Assume the generative process of equation 3 of the main text. $\mathbf{\Sigma_{11}}$ is $\mathrm{IW}_{n_1}(\nu, \mathbf{K}_{11})$ distributed for any principal submatrix of $\mathbf{\Sigma}$. Furthermore $y_1|\mathbf{\Sigma}_{11} \sim \mathrm{N}_{n_1}(0, (\nu-2)\mathbf{\Sigma}_{11})$ since the Gaussian distribution is consistent under marginalisation. Hence $y_1 \sim \mathrm{MVT}_{n_1}(\nu, \mu_1, \mathbf{K}_{11})$. □

We are now able to define a Student-t process.

Definition 6. *f is a Student-t process on $\mathcal{X}$ with parameters $\nu > 2$, mean function $\Phi : \mathcal{X} \to \mathbb{R}$, and kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ if any finite collection of function values have a joint multivariate Student-t distribution, i.e. $[f(x_1), ..., f(x_n)]^\top \sim \mathrm{MVT}_n(\nu, \boldsymbol{\phi}, \mathbf{K})$ where $\mathbf{K} \in \Pi(n)$ with $\mathbf{K}_{ij} = k(x_i, x_j)$ and $\phi \in \mathbb{R}^n$ with $\phi_i = \Phi(x_i)$. We write $f \sim \mathrm{TP}(\nu, \Phi, k)$.*

The Student-t process generalises the Gaussian process. In fact, a GP can be seen as a limiting case of a TP as shown in the following Lemma.

Lemma 7. *Suppose $f \sim \mathrm{TP}(\nu, \Phi, k)$ and $g \sim \mathrm{GP}(\Phi, k)$. Then f tends to g in distribution as $\nu \to \infty$.*

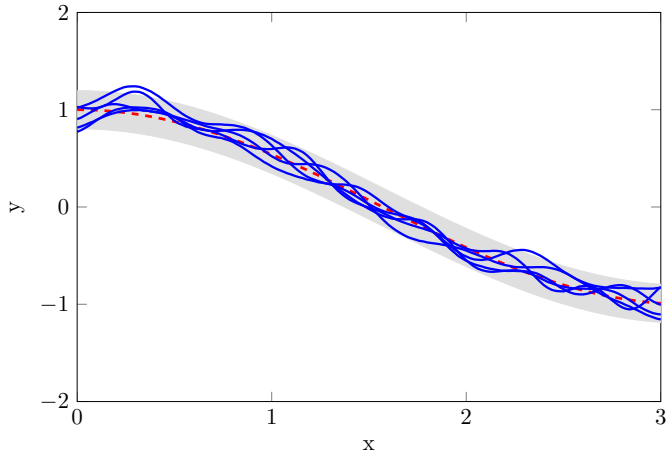
Proof. It is sufficient to show convergence in density for any finite collection of inputs. Let $\boldsymbol{y} \sim \mathrm{MVT}_n(\nu, \boldsymbol{\phi}, K)$ and set $\beta = (\boldsymbol{y} - \boldsymbol{\phi})^\top K^{-1}(\boldsymbol{y} - \boldsymbol{\phi})$ then as $\nu \to \infty$,

$$p(\boldsymbol{y}) \propto \left(1 + \frac{\beta}{\nu - 2}\right)^{-(\nu+n)/2} \to e^{-\beta/2}.$$

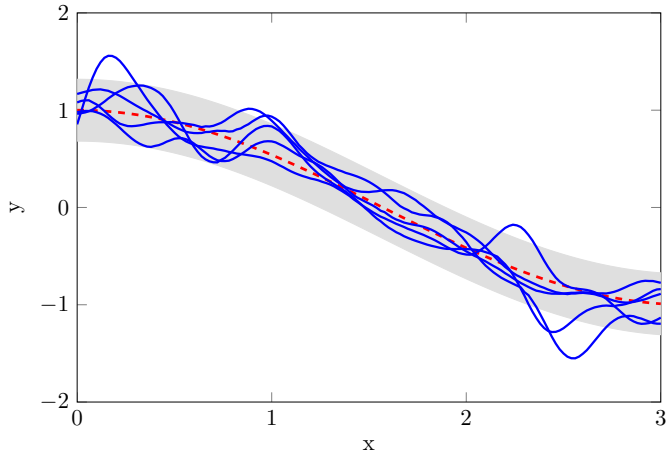
Hence the distribution of $\boldsymbol{y}$ tends to a $\mathrm{N}_n(\boldsymbol{\phi}, K)$ distribution as $\nu \to \infty$. □

The ν parameter controls how *heavy tailed* the Student-t process is. Smaller values of ν correspond to heavier tails. As ν gets larger, the tails converge to Gaussian tails. This is illustrated in prior sample draws shown in Figure 2.1. Notice that the samples from the TP tend to have more extreme behaviour than the GP, despite sharing the same mean and covariance functions.

ν also controls the dependence between variables which are jointly Student-t distributed, and not just their marginal distributions. In Figure 2.2 we show plots of samples which all have Gaussian marginals but different joint distributions determined by either a Student-t or a Gaussian copula. Note that in general, a bivariate distribution with Gaussian marginals but a



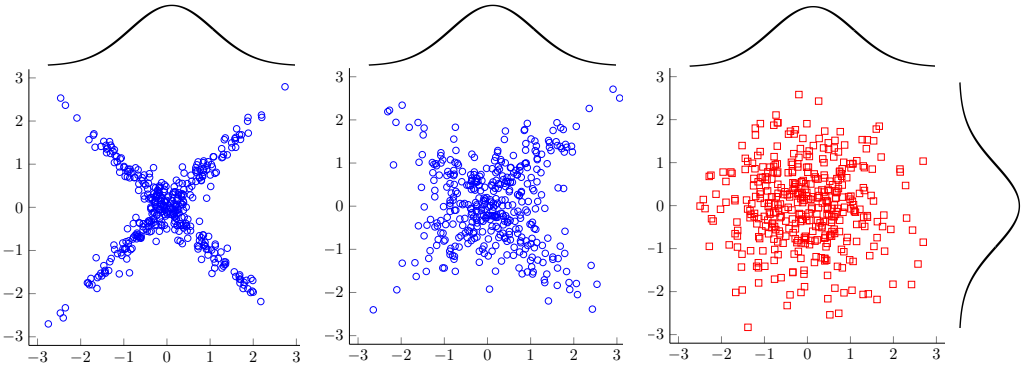
(a) Gaussian process samples



(b) Student-t process samples

Fig. 2.1 Five samples (blue solid) from (a) $\mathrm{GP}(\Phi, k)$ and (b) $\mathrm{TP}(\nu, \Phi, k)$, with $\nu = 3$. The mean and covariance functions are $\Phi(x) = \cos(x)$ (red dashed) and $k(x_i, x_j) = 0.01 \exp(-20(x_i - x_j)^2)$ respectively. The grey shaded area represents a 95% predictive interval under each model.

Student-t copula is *not* an elliptical distribution, hence the non-elliptical nature of Figure 2.2(a). Notice how the tail dependency of these distributions is controlled by ν. The dependency between $y(x_p)$ and $y(x_q)$ are different depending on whether y is a TP or a GP, even if they share the same covariance kernel.



(a) (b) (c)

Fig. 2.2 Uncorrelated bivariate samples from (a) a Student-t copula with $\nu = 3$, (b) a Student-t copula with $\nu = 10$ and (c) a Gaussian copula (right). All marginal distributions are $\mathrm{N}(0, 1)$ distributed.

The conditional distribution for a multivariate Student-t has an analytic form which we state and prove in Lemma 8.

Lemma 8. *Suppose* $\boldsymbol{y} \sim \mathrm{MVT}_n(\nu, \boldsymbol{\phi}, \mathbf{K})$ *and let* $\boldsymbol{y_1}$ *and* $\boldsymbol{y_2}$ *represent the first* n_1 *and remaining* n_2 *entries of* $\boldsymbol{y}$ *respectively. Then*

$$\boldsymbol{y_2}|\boldsymbol{y_1} \sim \mathrm{MVT}_{n_2}\left(\nu + n_1, \tilde{\boldsymbol{\phi}}_{\boldsymbol{2}}, \frac{\nu + \beta_1 - 2}{\nu + n_1 - 2} \times \tilde{\mathbf{K}}_{22}\right), \tag{2.7}$$

where

$$\begin{aligned} \tilde{\boldsymbol{\phi}}_{\boldsymbol{2}} &= \mathbf{K}_{21}\mathbf{K}_{11}^{-1}(\boldsymbol{y_1} - \boldsymbol{\phi_1}) + \boldsymbol{\phi_2} \\ \beta_1 &= (\boldsymbol{y_1} - \boldsymbol{\phi_1})^\top \mathbf{K}_{11}^{-1}(\boldsymbol{y_1} - \boldsymbol{\phi_1}), \\ \tilde{\mathbf{K}}_{22} &= K_{22} - \mathbf{K}_{21}\mathbf{K}_{11}^{-1}\mathbf{K}_{12}. \end{aligned}$$

Note that

$$\begin{aligned} \mathbb{E}[\boldsymbol{y_2}|\boldsymbol{y_1}] &= \tilde{\boldsymbol{\phi}}_{\boldsymbol{2}}, \\ \mathrm{cov}[\boldsymbol{y_2}|\boldsymbol{y_1}] &= \frac{\nu + \beta_1 - 2}{\nu + n_1 - 2} \times \tilde{\mathbf{K}}_{22}. \end{aligned}$$

Proof. Let $\beta_2 = (\boldsymbol{y_2} - \tilde{\boldsymbol{\phi}}_{\boldsymbol{2}})^\top \tilde{K}_{22}^{-1}(\boldsymbol{y_2} - \tilde{\boldsymbol{\phi}}_{\boldsymbol{2}})$.
Note that $\beta_1 + \beta_2 = (\boldsymbol{y} - \boldsymbol{\phi})^\top K^{-1}(\boldsymbol{y} - \boldsymbol{\phi})$. We have

$$p(\boldsymbol{y_2}|\boldsymbol{y_1}) = \frac{p(\boldsymbol{y_1}, \boldsymbol{y_2})}{p(\boldsymbol{y_1})} \propto \left(1 + \frac{\beta_1 + \beta_2}{\nu - 2}\right)^{-(\nu+n)/2} \left(1 + \frac{\beta_1}{\nu - 2}\right)^{(\nu+n_1)/2}$$
$$\propto \left(1 + \frac{\beta_2}{\beta_1 + \nu - 2}\right)^{-(\nu+n)/2}$$

Comparing this expression to the definition of a MVT density function gives the required result. □

As ν tends to infinity, this predictive distribution tends to a Gaussian process predictive distribution as we would expect given Lemma 7. Perhaps less intuitively, this predictive distribution also tends to a Gaussian process predictive as n_1 tends to infinity. For n_1 above 50, there typically would not be a noticeable difference between the Student-t and Gaussian predictive distribution. Therefore the Student-t process is interesting predominantly in the low data setting.

The predictive mean has the same form as for a Gaussian process, conditioned on having the same kernel k, with the same hyperparameters. The key difference is in the predictive covariance, which now explicitly depends on the training observations. Indeed, a somewhat disappointing feature of the Gaussian process is that for a given kernel with fixed hyperparameters, the predictive covariance of new samples does not depend on training observations. Importantly, since the marginal likelihood of the TP in (2.5) differs from the marginal likelihood of the GP, both the predictive mean and predictive covariance of a TP will differ from that of a GP, after learning kernel hyperparameters.

The scaling constant of the multivariate Student-t predictive covariance has an intuitive explanation. Note that β_1 is distributed as the sum of squares of n_1 independent $\mathrm{MVT}_1(\nu, 0, 1)$ distributions and hence $\mathbb{E}[\beta_1] = n_1$. If the observed value of β_1 is larger than n_1, the predictive covariance is scaled up, and the converse holds also. The magnitude of scaling is controlled by ν.

2.4 Elliptical Processes

Having derived the Student-t process and discussed some of its properties, in this section we consider the more general class of elliptical processes. We begin by motivating and describing an elliptical distribution.

Gaussian distributions are a popular choice for statistical modelling across a broad range of sciences in part because its density has elliptical contours centred around a single mode. This encodes the belief that a quantity has one most likely value, and the probability density decreases as we move away from this mode. It is worth asking the question, "what is the class of distributions that are unimodal with elliptical contours?" Appropriately, the answer is, *elliptical distributions*. We define an elliptical distribution based on an equivalency result from Cambanis et al. [1981].

Definition 9. *$\boldsymbol{y} \in \mathbb{R}^n$ follows an elliptical distribution if and only if there exists constant $\boldsymbol{\mu} \in \mathbb{R}^n$, constant $n \times d$ matrix $\boldsymbol{\Omega}$ with maximal rank d, non-negative random scalar R and $\boldsymbol{u}$ independent random which is uniformly distributed on the unit sphere in $\mathbb{R}^d$, such that $\boldsymbol{y} \overset{\mathcal{D}}{=} \boldsymbol{\mu} + R\boldsymbol{\Omega}\boldsymbol{u}$, where $\overset{\mathcal{D}}{=}$ denotes equality in distribution.*

An overview of properties of elliptical distributions can be found at Fang et al. [1989], along with the proof of the following lemma.

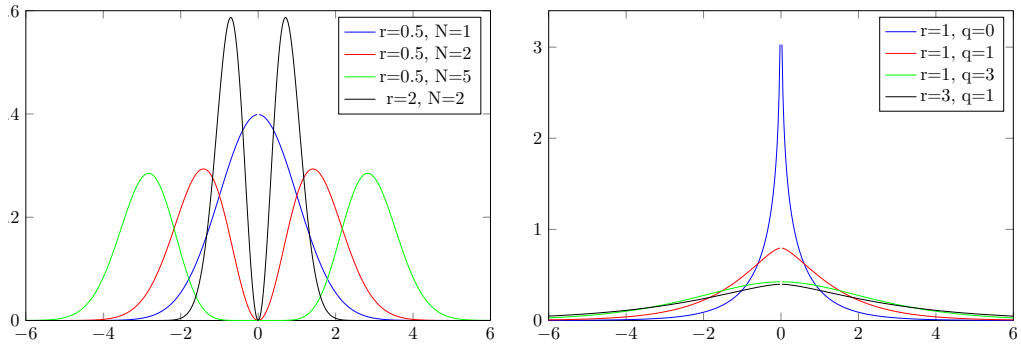
Lemma 10. *Suppose $R_1 \sim \chi^2(n)$ and $R_2 \sim \Gamma^{-1}(\nu/2, 1/2)$ independently. If $R \overset{\mathcal{D}}{=} \sqrt{R_1}$, then $\boldsymbol{y}$ is Gaussian distributed. If $R \overset{\mathcal{D}}{=} \sqrt{(\nu-2)R_1R_2}$ then $\boldsymbol{y}$ is MVT distributed.*

The result above motivates an alternative way to generate a multivariate Student-t distribution from a Gaussian distribution, which we prove in the following lemma.

Lemma 11. *Let $\mathbf{K} \in \Pi(n)$, $\boldsymbol{\phi} \in \mathbb{R}^n$, $\nu > 2$, $\rho > 0$, all constant, and*

$$\begin{aligned} r^{-1} &\sim \Gamma(\nu/2, \rho/2) \\ \boldsymbol{y}|r &\sim \mathrm{N}_n(\boldsymbol{\phi}, r(\nu-2)\mathbf{K}/\rho), \end{aligned} \tag{2.8}$$

then marginally $\boldsymbol{y} \sim \mathrm{MVT}_n(\nu, \boldsymbol{\phi}, \mathbf{K})$.



(a) Kotz density (b) Bessel density

Fig. 2.3 Density plots for elliptical distributions in 1 dimension including (a) Kotz and (b) Bessel densities with various parameters.

Proof. Let $\beta = (\boldsymbol{y} - \boldsymbol{\phi})^\top \mathbf{K}^{-1}(\boldsymbol{y} - \boldsymbol{\phi})$. We can analytically marginalise out the scalar r,

$$
\begin{aligned}
p(\boldsymbol{y}) = \int p(\boldsymbol{y}|r)p(r)dr &\propto \int \exp\Big(-\frac{\rho\beta}{2(\nu-2)r}\Big) r^{-\frac{n}{2}} \exp\Big(-\frac{\rho}{2r}\Big) r^{-\frac{(\nu+2)}{2}} dr \\
&\propto \Big(1 + \frac{\beta}{\nu-2}\Big)^{-\frac{(\nu+n)}{2}} \int \exp\Big(-\frac{1}{2r}\Big) r^{-\frac{(\nu+n+2)}{2}} dr \\
&\propto \Big(1 + \frac{\beta}{\nu-2}\Big)^{-\frac{(\nu+n)}{2}}
\end{aligned}
$$

Hence $\boldsymbol{y} \sim \mathrm{MVT}_n(\nu, \boldsymbol{\phi}, \mathbf{K})$. Note the redundancy in ρ. Without loss of generality, let $\rho = 1$. □

This is a surprising finding because previously, in (2.4), we derived a Student-t process by placing an IWP prior on the covariance matrix, which appears to be far more rich a distribution than a deterministic covariance function scaled by an inverse Gamma random variable, yet they both lead to the Student-t process! In the next section, we return to this idea, and use it to motivate a new way to construct the IWP, shedding light on some of its properties.

Below are a couple of elliptical distributions and their probability density functions other than the multivariate Gaussian and Student-t. See Figure 2.3 for plots of their densities.

Example 12. *The Kotz distribution,* $\boldsymbol{y} \sim \mathrm{K}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}, N, r)$ *for* $N > \frac{2-n}{2}$ *and* $r > 0$ *has density*

$$p(\boldsymbol{y}) = \frac{\Gamma\left(\frac{n}{2}\right) r^{\frac{2N-2+n}{2}}}{\Gamma\left(\frac{2N-2+n}{2}\right) \pi^{\frac{n}{2}} |\Sigma|^{\frac{n}{2}}} \beta^{N-1} \exp(-r\beta),$$

for $\beta = (\boldsymbol{y} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{y} - \boldsymbol{\mu})$.

Example 13. *The Bessel distribution,* $\boldsymbol{y} \sim \mathrm{B}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}, q, r)$ *for* $q > -\frac{n}{2}$ *and* $r > 0$ *has density*

$$p(\boldsymbol{y}) = \frac{1}{2^{q+n-1} \pi^{\frac{n}{2}} r^{n+q} \Gamma\left(q + \frac{n}{2}\right) |\Sigma|^{\frac{n}{2}}} \beta^{\frac{q}{2}} K_q\left(\frac{\sqrt{\beta}}{r}\right),$$

for $\beta = (\boldsymbol{y} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{y} - \boldsymbol{\mu})$ *and*

$$K_q(z) = \frac{\pi}{2 \sin(q\pi)} (I_{-q}(z) - I_q(z))$$
$$I_q(z) = \sum_{k=0}^{\infty} \frac{1}{k! \Gamma(k + q + 1)} \left(\frac{z}{2}\right)^{q+2k}.$$

Elliptical distributions characterise a large class of distributions which are unimodal and where the likelihood of a point decreases in its distance from this mode. The idea naturally extends to infinite dimensional objects.

Definition 14. *Let* $\mathcal{Y} = \{y_i\}$ *be a countable family of random variables. It is an elliptical process if any finite subset of them jointly have an elliptical distribution.*

Not all elliptical distributions have densities (e.g. Lévy, alpha-stable distributions). Even fewer elliptical processes have densities, and the set of those that do is characterised in Theorem 15 due to Kelker [1970].

Theorem 15. *Suppose* $\mathcal{Y} = \{y_i\}$ *is an elliptical process. Any finite collection* $\boldsymbol{z} = \{z_1, ..., z_n\} \subset \mathcal{Y}$ *has a density if and only if there exists a non-negative random variable* r *such that* $\boldsymbol{z}|r \sim \mathrm{N}_n(\boldsymbol{\mu}, r\boldsymbol{\Omega}\boldsymbol{\Omega}^\top)$.

Rather surprisingly, whilst there exist a vast range of elliptical distributions, the theorem above suggests that the class of elliptical processes is significantly

more limited. Scale mixtures of Gaussian distributions have been successfully used for machine learning in the past [Wainwright and Simoncelli, 1999; Portilla et al., 2003]. A simple corollary of the theorem describes the only two cases where an elliptical process has an analytically representable density function.

Corollary 16. *Suppose $\mathcal{Y} = \{y_i\}$ is an elliptical process. Any finite collection $\boldsymbol{z} = \{z_1, ..., z_n\} \subset \mathcal{Y}$ has an analytically representable density if and only if $\mathcal{Y}$ is either a Gaussian process or a Student-t process.*

Proof. By Theorem 15, we need to be able to analytically solve $\int p(\boldsymbol{z}|r)p(r)dr$, where $\boldsymbol{z}|r \sim \mathrm{N}_n(\boldsymbol{\mu}, r\boldsymbol{\Omega}\boldsymbol{\Omega}^\top)$. This is possible either when r is a constant with probability 1 or when $r \sim \Gamma^{-1}(\nu/2, 1/2)$, the conjugate prior. These lead to the Gaussian and Student-t processes respectively. □

Since the Student-t process generalises the Gaussian process, it is the most general elliptical process which has an analytically representable density. The TP is thus an expressive tool for non-parametric Bayesian modelling. We will return to applying the Student-t process to a range of regression and Bayesian optimisation tasks in the experiments section, but first, we take a slight detour into understanding the inverse Wishart process.

2.5 Deconstructing the Inverse Wishart Distribution

We show that the density of an inverse Wishart distribution depends only on the eigenvalues of a positive definite matrix. To the best of our knowledge this change of variables has not been computed previously. This decomposition offers a novel way of sampling from an inverse Wishart distribution and insight into why the Student-t process can be derived using either an inverse Gamma, or an inverse Wishart process covariance prior.

Let $\Xi(n)$ be the set of all $n \times n$ orthogonal matrices. A matrix is orthogonal if it is square, real valued and its rows and columns are orthogonal unit vectors. Orthogonal matrices are compositions of rotations and reflections,

which are volume preserving operations. Symmetric positive definite (SPD) matrices can be represented through a diagonal and an orthogonal matrix:

Theorem 17. *Let* $\mathbf{\Sigma} \in \Pi(n)$*, the set of SPD,* $n \times n$ *matrices. Suppose* $\{\lambda_1, ..., \lambda_n\}$ *are the eigenvalues of* $\mathbf{\Sigma}$*. There exists* $\boldsymbol{Q} \in \Xi(n)$ *such that* $\mathbf{\Sigma} = \boldsymbol{Q}\mathbf{\Lambda}\boldsymbol{Q}^\top$*, where* $\mathbf{\Lambda} = \text{diag}(\lambda_1, ..., \lambda_n)$*.*

Now suppose $\mathbf{\Sigma} \sim \text{IW}_n(\nu, \mathbf{I})$. We compute the density of an IW using the representation in Theorem 17, being careful to include the Jacobian of the change of variable, $J(\mathbf{\Sigma}; \mathbf{Q}, \mathbf{\Lambda})$, given in Edelman and Rao [2005]. From (2.2) and using the facts that $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$ and $|\mathbf{AB}| = |\mathbf{BA}|$,

$$\begin{aligned}
p(\mathbf{\Sigma})d\mathbf{\Sigma} &= p(\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top)|J(\mathbf{\Sigma}; \mathbf{Q}, \mathbf{\Lambda})|d\mathbf{\Lambda}d\mathbf{Q} \\
&\propto |\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top|^{-(\nu+2n)/2} \exp\Big(-\frac{1}{2}\text{Tr}\Big((\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top)^{-1}\Big)\Big) \\
&\qquad \times \Big|\mathbf{Q}^\top \prod_{1\le i<j\le n} |\lambda_i - \lambda_j|\Big| d\mathbf{\Lambda}d\mathbf{Q} \\
&\propto |\mathbf{\Lambda}|^{-(\nu+2n)/2} \exp\Big(-\frac{1}{2}\text{Tr}\Big(\mathbf{\Lambda}^{-1}\Big)\Big) \prod_{1\le i<j\le n} |\lambda_i - \lambda_j|^n d\mathbf{\Lambda}d\mathbf{Q} \\
&\propto \prod_{i=1}^{n} \left(\lambda_i^{-\frac{\nu+2n}{2}} e^{-\frac{1}{2\lambda_i}} \prod_{j\neq i} |\lambda_i - \lambda_j|^n d\lambda_i\right) d\mathbf{Q}. \qquad (2.9)
\end{aligned}$$

We see from (2.9) that $\mathbf{Q}$ is uniformly distributed over $\Xi(n)$ (e.g. from a $\Upsilon_{n,n}$ distribution as described in Dawid [1977]) and that the λ_i are exchangeable, i.e., permuting the $\text{diag}(\mathbf{\Lambda})$ does not affect its probability. We denote this exchangeable distribution $\Theta_n(\nu)$. We generate a draw from an inverse Wishart distribution by sampling $\mathbf{Q} \sim \Upsilon_{n,n}$, $\mathbf{\Lambda} \sim \Theta_n(\nu)$ and setting $\mathbf{\Sigma} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$.

This result provides a geometric interpretation of what a sample from $\text{IW}_n(\nu, \mathbf{I})$ looks like. We first uniformly at random pick an orthogonal set of basis vectors in $\mathbb{R}^n$ and then stretch these basis vectors using an exchangeable set of scalar random variables. An analogous interpretation holds for the Wishart distribution.

Finally, the generative process above gives us useful intuition into why using an inverse Wishart process and an inverse Gamma scaled covariance

prior both lead to the same Student-t process marginal distribution.

Recall from Lemma 10 that if $\boldsymbol{u}$ is uniformly distributed on the unit sphere in $\mathbb{R}^n$ and $R \sim \chi^2(n)$ independently, then $\sqrt{R}\boldsymbol{u} \sim \mathrm{N}_n(0, \mathbf{I})$. By (2.4) and Lemma 10, if we sample $\mathbf{Q}$ and $\boldsymbol{\Lambda}$ from the generative process above, then $\sqrt{(\nu-2)R}\mathbf{Q}\boldsymbol{\Lambda}^{1/2}\boldsymbol{u}$ is marginally a draw from $\mathrm{MVT}(\nu, 0, \mathbf{I})$. Since the diagonal elements of $\boldsymbol{\Lambda}$ are exchangeable, $\mathbf{Q}$ is orthogonal and sampled uniformly over $\Xi(n)$, and $\boldsymbol{u}$ is spherically symmetric, composing the actions of $\mathbf{Q}$ and $\boldsymbol{\Lambda}^{1/2}$ on $\boldsymbol{u}$ is equivalent to that of a scalar random variable. Concretely, we must have that $Q\boldsymbol{\Lambda}^{1/2}\boldsymbol{u} \overset{\mathcal{D}}{=} \sqrt{R'}\boldsymbol{u}$ for some positive scalar random variable R', by symmetry. By Lemma 10 we know $R' \sim \Gamma^{-1}(\nu/2, 1/2)$. In summary, the action of $\mathbf{Q}\boldsymbol{\Lambda}^{1/2}$ on $\boldsymbol{u}$ is equivalent in distribution to a rescaling by an inverse Gamma variate.

2.6 Applications

We illustrate the benefits of a Student-t process model in regression and Bayesian optimisation tasks in this section.

2.6.1 Regression

Consider a set of observations $\{x_i, y_i\}_{i=1}^n$ for $x_i \in \mathcal{X}$ and $y_i \in \mathbb{R}$. Analogous to Gaussian process regression, we assume the following generative model

$$\begin{aligned} f &\sim \mathrm{TP}(\nu, \Phi, k_\theta) \\ y_i &= f(x_i) \quad \text{for } i = 1, ..., n. \end{aligned} \tag{2.10}$$

In this work we consider parametric kernel functions. A key task when using such kernels is in learning the parameters of the chosen kernel, which are called the hyperparameters of the model, as described in the introduction to Gaussian processes. The derivatives of the marginal log likelihood of the TP with respect to the hyperparameters are typically used for gradient based optimisation or sampling.

TP Marginal Likelihood Derivatives

Suppose $\boldsymbol{y} \sim \mathrm{MVT}_n(\nu, \boldsymbol{\phi}, \mathbf{K}_\theta)$, then

$$\log p(\boldsymbol{y}|\nu, \boldsymbol{\phi}, \mathbf{K}_\theta) = -\frac{n}{2}\log((\nu-2)\pi) + \log\left(\frac{\Gamma(\frac{\nu+n}{2})}{\Gamma(\frac{\nu}{2})}\right) - \frac{1}{2}\log(|\mathbf{K}_\theta|) - \frac{(\nu+n)}{2}\log\left(1+\frac{\beta}{\nu-2}\right), \tag{2.11}$$

where $\beta = (\boldsymbol{y}-\boldsymbol{\phi})^\top \mathbf{K}_\theta^{-1}(\boldsymbol{y}-\boldsymbol{\phi})$ and its derivative with respect to hyperparameter θ is

$$\frac{\partial}{\partial\theta}\log p(\boldsymbol{y}|\nu, \boldsymbol{\phi}, \mathbf{K}_\theta) = \frac{1}{2}\mathrm{Tr}\left(\left(\frac{\nu+n}{\nu+\beta-2}\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - \mathbf{K}_\theta^{-1}\right)\frac{\partial \mathbf{K}_\theta}{\partial\theta}\right),$$

where $\boldsymbol{\alpha} = \mathbf{K}_\theta^{-1}(\boldsymbol{y}-\boldsymbol{\phi})$. We may also learn ν using gradient based methods and the following derivative

$$\frac{\partial}{\partial\nu}\log p(\boldsymbol{y}|\nu, \mathbf{K}_\theta) = -\frac{n}{2(\nu-2)} + \psi\left(\frac{\nu+n}{2}\right) - \psi\left(\frac{\nu}{2}\right) - \frac{1}{2}\log\left(1+\frac{\beta}{\nu-2}\right) + \frac{(\nu+n)\beta}{2(\nu-2)^2+2\beta(\nu-2)} \tag{2.12}$$

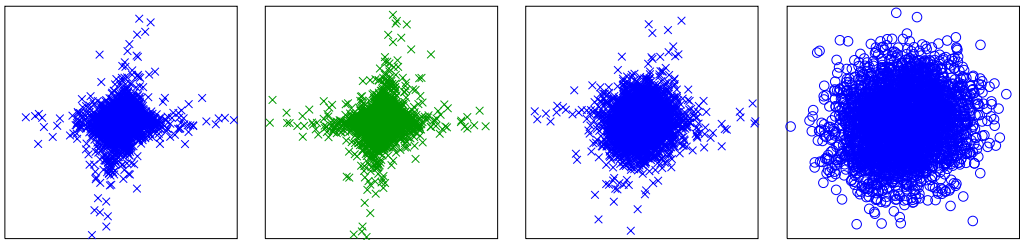
where ψ is the digamma function.

It is also common to assume that the observations, y_i, are actually noisy versions of the function values, f_i. We briefly describe our approach to incorporating function noise into our model.

Incorporating Observation Noise

It is common practice to assume that outputs are the sum of a latent Gaussian process and independent Gaussian noise. An advantage of such a model is in the fact that the sum of independent Gaussian distributions is Gaussian distributed and hence such a Gaussian process model remains analytic in the presence of noise. Unfortunately the sum of two independent MVTs is analytically intractable.

This problem was encountered by Rasmussen and Williams [2006], who



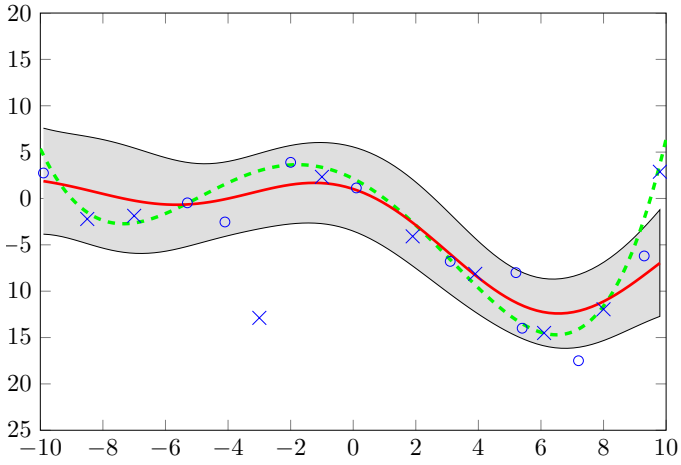
(a) (b) (c) (d)

Fig. 2.4 Scatter plots of points drawn from various 2-dim processes. Here $\nu = 2.1$ and $\mathbf{K}_{ij} = 0.8\delta_{ij} + 0.2$. (a) Samples from $x+y$, where $x \sim \mathrm{MVT}_2(\nu, 0, \mathbf{K})$, $y \sim \mathrm{MVT}_2(\nu, 0, 0.5\mathbf{I})$, (b) samples from x, where $x \sim \mathrm{MVT}_2(\nu, 0, \mathbf{K} + 0.5\mathbf{I})$ (our model), (c) samples from $x + y$, where $x \sim \mathrm{MVT}_2(\nu, 0, \mathbf{K})$, $y \sim \mathrm{N}_2(0, 0.5\mathbf{I})$, (d) samples from x, where $x \sim \mathrm{N}_2(0, \mathbf{K} + 0.5\mathbf{I})$.

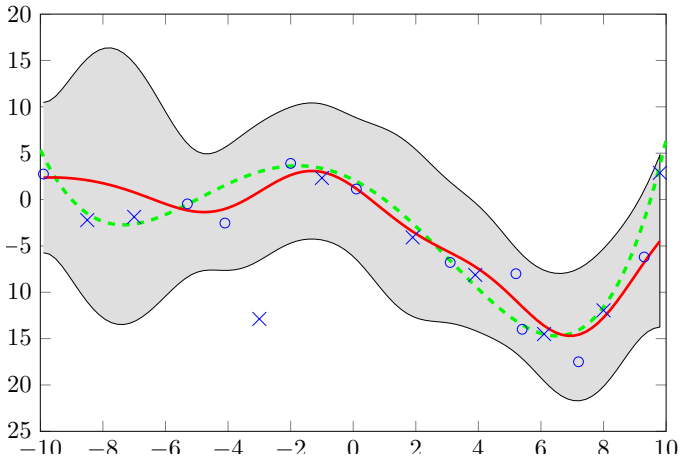
went on to dismiss the multivariate Student-t process for practical purposes. Our approach is to incorporate the noise into the kernel function, for example, letting $k = k_\theta + \sigma^2\delta$, where k_θ is a parametrised kernel and δ is a diagonal kernel function. Such a model is not equivalent to adding independent noise, since the scaling parameter ν will have an effect on the squared-exponential kernel as well as the noise kernel. Zhang and Yeung [2010] propose a similar method for handling noise; however, they incorrectly assume that the latent function and noise are independent under this model. The noise will be uncorrelated with the latent function, but not independent.

As $\nu \to \infty$ this model tends to a GP with independent Gaussian noise, this is a simple consequence of Lemma 7. In Figure 2.4, we consider samples from various two dimensional processes when ν is small and the signal to noise ratio is small. Here we see that the MVT with noise incorporated into its kernel behaves similarly to a TP with independent Student-t noise.

There have been several attempts to make GP regression robust to heavy tailed noise that rely on approximate inference [Neal, 1997; Vanhatalo et al., 2009]. It is hence attractive that our proposed method can model heavy tailed noise whilst retaining an analytic inference scheme. This is a novel finding to the best of our knowledge.



(a) Gaussian process posterior



(b) Student-t process posterior

Fig. 2.5 Posterior distributions of 1 sample from Synthetic Data B under (a) GP prior and (b) TP prior. The dotted green line is the true function without noise, the solid red line is the posterior mean, the shaded area represents a 95% posterior predictive interval, circles are training points and crosses are test points.

Experiments

We test the Student-t process as a regression model on a number of datasets. We sample hyperparameters using Hamiltonian Monte Carlo [Neal, 2011] and use a kernel function which is a sum of a squared exponential and a delta kernel function ($k_\theta = k_{\text{SE}}$). The results for all of these experiments are

Table 2.1 Predictive Mean Squared Errors (MSE) and Log Likelihoods (LL) of regression experiments. The TP consistently has the lowest MSE and highest LL.

Data set	Gaussian Process		Student-T Process	
	MSE	LL	MSE	LL
Synth A	2.24 ± 0.09	-1.66± 0.04	2.29 ± 0.08	**-1.00± 0.03**
Synth B	9.53 ± 0.03	-1.45± 0.02	**5.69 ± 0.03**	**-1.30± 0.02**
Snow	10.2 ± 0.08	4.00 ± 0.12	10.5 ± 0.07	**25.7 ± 0.18**
Spatial	6.89 ±0.04	4.34±0.22	**5.71 ±0.03**	**44.4±0.4**
Wine	4.84 ± 0.08	-1.4 ± 1	**4.20 ± 0.06**	**113 ± 2**

summarised in Table 2.1.

Synthetic Data A. We sample 100 functions from a GP prior with Gaussian noise and fit both GPs and TPs to the data with the goal of predicting test points. For each function we train on 80 data points and test on 20. The TP, which generalises the GP, has superior predictive uncertainty in this example.

Synthetic Data B. We construct data by drawing 100 functions from a GP with a squared exponential kernel and adding Student-t noise independently. The posterior distribution of one sample is shown in Figure 2.5. The predictive means are also not identical since the posterior distributions of the hyperparameters differ between the TP and the GP, due to their different likelihoods. This is why placing priors over Gaussian process hyperparameters does not quite achieve the same effect as using a Student-t process model. Here the TP has a superior predictive mean, since after hyperparameter training it is better able to model Student-t noise. It also seems like the GP posterior is too confident about some of its predictions versus the TP e.g. for large positive input values on the right hand side of the plots.

Whistler Snowfall Data[1]**.** Daily snowfall amounts in Whistler have been

[1]The snowfall dataset can be found at http://www.climate.weatheroffice.ec.gc.ca.

recorded for the years 2010 and 2011. This data exhibits clear change-point type behaviour due to seasonality which the TP handles much better than the GP.

Spatial Interpolation Data[2]**.** This dataset contains rainfall measurements at 467 (100 observed and 367 for testing) locations in Switzerland on 8 May 1986.

Wine Data. This dataset due to Cortez et al. [2009] consists of 12 attributes of various red wines including acidity, density, pH and alcohol level. Each wine is given a corresponding quality score between 0 and 10. We choose a random subset of 400 wines: 360 for training and 40 for testing.

The predictive ability of the Student-t process appears to be at least as good as that of the Gaussian process amongst all of the regression tasks we have explored, and is significantly better when the noise is heavy tailed, or when there is any heteroskedastic unusual behaviour in the underlying process.

2.6.2 Bayesian Optimisation

The Student-t process often ends up having a similar predictive mean to that of a Gaussian process model, however, they tend to differ in their predictive uncertainty. This is in particular due to the ν parameter which controls the tail-dependence of the TP. We were curious to apply the Student-t process to a task where predictive uncertainty played a key role, and Bayesian optimisation appeared to be the ideal candidate.

Method

In this work, we chose the expected improvement (EI) acquisition function, and for reasons described in Snoek et al. [2012], used an ARD Matérn 5/2

[2]The spatial interpolation data can be found at http://www.ai_geostats.org under SIC97.

kernel defined as

$$k_{M52}(\boldsymbol{x}, \boldsymbol{x}') = \theta_0\Big(1 + \sqrt{5r^2_{\boldsymbol{x},\boldsymbol{x}'}}\Big) \exp\Big(- \sqrt{5r^2_{\boldsymbol{x},\boldsymbol{x}'}}\Big) \tag{2.13}$$

where $r^2(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\theta_d^2}$.

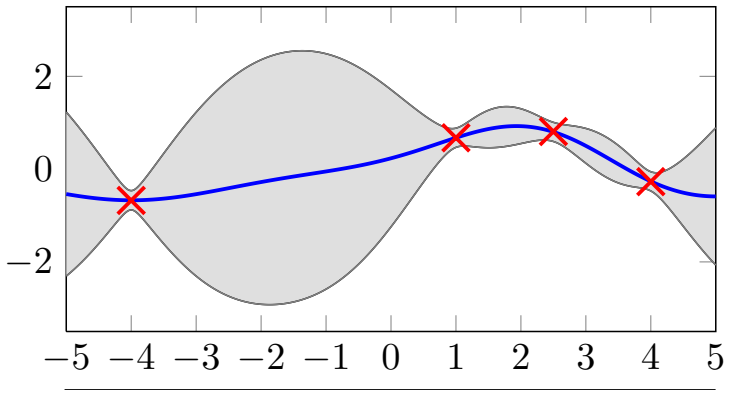
We assume that the function we wish to optimise over is $f : \mathbb{R}^D \to \mathbb{R}$ and is a draw from a multivariate Student-t process with scale parameter $\nu > 2$, constant mean μ and kernel function of the form $k_{M52}(\boldsymbol{x}, \boldsymbol{x}') + \sigma^2\delta(\boldsymbol{x}, \boldsymbol{x}')$.

Our goal is to find where f attains its minimum. Let $\mathcal{D} = \{\boldsymbol{x}_n, f_n\}_{n=1}^N$ be our current set of N observations and $f_{\text{best}} = \min\{f_1, ..., f_N\}$. To compress notation we let $\boldsymbol{\theta}$ represent the parameters θ, ν, μ. Let the acquisition function $a_{\text{EI}}\Big(\boldsymbol{x}; \mathcal{D}, \boldsymbol{\theta}\Big)$ denote the expected improvement over the current best value from choosing to sample at point $\boldsymbol{x}$ given current observations $\mathcal{D}$ and hyperparameters $\boldsymbol{\theta}$. Note that the distribution of $f(\boldsymbol{x})|\mathcal{D}, \boldsymbol{\theta}$ is $\text{MVT}_1(\nu + N, \tilde{\mu}(\boldsymbol{x}; X_n), \tilde{\tau}(\boldsymbol{x}; X_n, \nu)^2)$, where the form of $\tilde{\mu}$ and $\tilde{\tau}$ are derived in Lemma 8. Let $\tilde{\gamma} = \frac{f_{\text{best}} - \tilde{\mu}}{\tilde{\tau}}$. Then

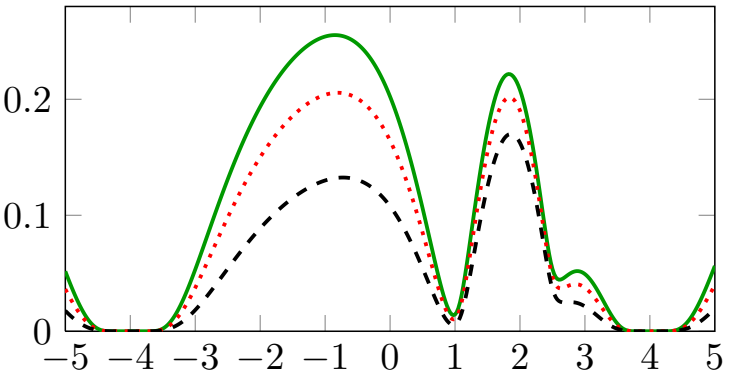
$$\begin{aligned}
a_{\text{EI}}\Big(\boldsymbol{x}; \mathcal{D}, \boldsymbol{\theta}\Big) &= \mathbb{E}\Big[\max\Big(f_{\text{best}} - f(\boldsymbol{x}), 0\Big)|\mathcal{D}, \boldsymbol{\theta}\Big] \\
&= \int_{-\infty}^{f_{\text{best}}} dy (f_{\text{best}} - y) \frac{1}{\tilde{\tau}} \lambda_{\nu+N}\Big(\frac{y - \tilde{\mu}}{\tilde{\tau}}\Big) \\
&= \int_{-\infty}^{\tilde{\gamma}} dy (f_{\text{best}} - \tilde{\tau}y - \tilde{\mu}) \lambda_{\nu+N}(y) \\
&= \tilde{\gamma}\tilde{\tau}\Lambda_{\nu+N}(\tilde{\gamma}) + \tilde{\tau}\Big(1 + \frac{\tilde{\gamma}^2 - 1}{\nu + N - 1}\Big)\lambda_{\nu+N}(\tilde{\gamma}),
\end{aligned} \tag{2.14}$$

where λ_ν and Λ_ν are the density and distribution functions of a $\text{MVT}_1(\nu, 0, 1)$ distribution respectively.

We treat the hyperparameters in a Bayesian fashion, placing vague gamma priors on θ, σ and $\nu - 2$, and a Gaussian prior on μ. The hyperparameters are all sampled from the posterior using slice sampling [Neal, 2003]. Suppose



(a) Gaussian process posterior



(b) Student-t Expected improvement varying ν

Fig. 2.6 (a) Posterior distribution of a function to maximise under a GP prior, red crosses at observed values, blue line for the predictive mean and grey shade for the 95 % predictive interval. (b) Expected improvement acquisition functions: the solid green line is the acquisition function for a GP, the dotted red and dashed black lines are for TP priors with $\nu = 15$ and $\nu = 5$ respectively. All other hyperparameters are kept the same.

we have H sets of posterior samples $\{\boldsymbol{\theta}_h\}_{h=1}^{H}$. We set

$$\tilde{a}_{\mathrm{EI}}\Big(\boldsymbol{x}; \mathcal{D}\Big) = \frac{1}{H} \sum_{h=1}^{H} a_{\mathrm{EI}}\Big(\boldsymbol{x}; \mathcal{D}, \boldsymbol{\theta}_h\Big) \tag{2.15}$$

as our approximate marginalised acquisition function. The next evaluation takes place at $\boldsymbol{x}_{\mathrm{next}} = \mathrm{argmax}_{\boldsymbol{x} \in \mathbb{R}^D} \, \tilde{a}_{\mathrm{EI}}\Big(\boldsymbol{x}; \mathcal{D}\Big)$, found using gradient descent based methods starting from a dense set of points in the input space.

To get more intuition on how ν changes the behaviour of the acquisition function, we study an example in Figure 2.6. Here we fix all hyperparameters other than ν and plot the acquisition functions varying ν. In this example, it is clear that in certain scenarios the TP prior and GP prior will lead to very different proposals given the same information. The mode of the expected improvement acquisition function can drastically change as ν changes.

Experiments

We compare a TP prior with a Matérn plus a delta function kernel to a GP prior with the same kernel with Monte Carlo integration over the hyperparameters, for Bayesian optimisation. We consider 3 functions: a 1-dim synthetic sinusoidal, the 2-dim Branin-Hoo function and a 6-dim Hartmann function. All the results are shown in Figure 2.7.

Sinusoidal synthetic function In this experiment we aimed to find the minimum of $f(x) = -(x-1)^2 \sin(3x + 5x^{-1} + 1)$ in the interval $[5, 10]$. The function has 2 local minima in this interval. TP optimisation clearly outperforms GP optimization in this problem; the TP was able to come to within 0.1% of the minimum in 8.1 ± 0.4 iterations whilst the GP took 10.7 ± 0.6 iterations.

Branin-Hoo function This function is a popular benchmark for optimisation methods [Jones, 2001a] and is defined on the set $\{(x_1, x_2) : 0 \leq x_1 \leq 15, -5 \leq x_2 \leq 15\}$. We initialised the runs with 4 initial observations, one for each corner of the input square.

Hartmann function This is a function with 6 local minima in $[0, 1]^6$ [Picheny et al., 2013]. The runs are initialised with 6 observations at randomly chosen corners of the unit cube in $\mathbb{R}^6$.

The performance of Bayesian optimisation with the Student-t process is consistently better than that of the Gaussian process (Figure 2.7). The key driver of the difference in performance is the control of the higher order

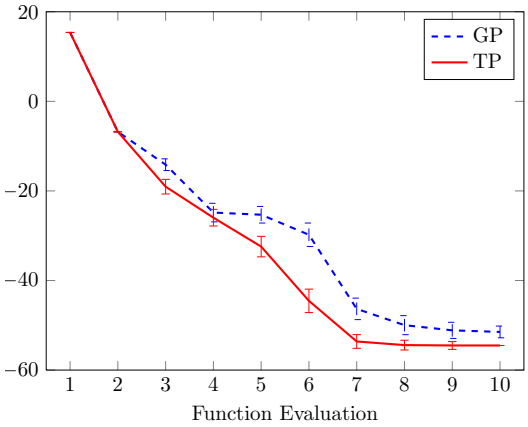
moments of the process due to the ν parameter, which the Gaussian process does not have the ability to do.

2.7 Conclusions

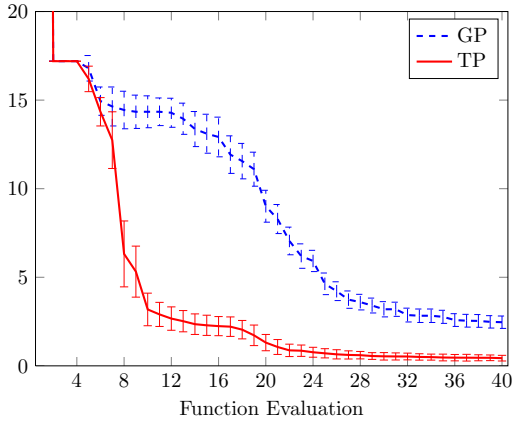
We have shown that the inverse Wishart process (IWP) is an appropriate prior over covariance matrices of arbitrary size. We used an IWP prior over a GP kernel and showed that marginalising over the IWP results in a Student-t process (TP). The TP has consistent marginals, closed form conditionals and contains the Gaussian process as a special case. We also proved that the TP is the only elliptical process other than the GP which has an analytically representable density function. The TP prior was applied in regression and Bayesian optimisation tasks, showing improved performance over GPs with no additional computational costs, other than sampling or optimising ν.

The take home message for practitioners should be that the TP has many if not all of the benefits of GPs, but with increased modelling flexibility at negligible extra cost. Our work suggests that it could be useful to replace GPs with TPs in almost any application. The added flexibility of the TP is orthogonal to the choice of kernel, and could complement recent expressive closed form kernels [Wilson and Adams, 2013; Wilson et al., 2013] in future work.

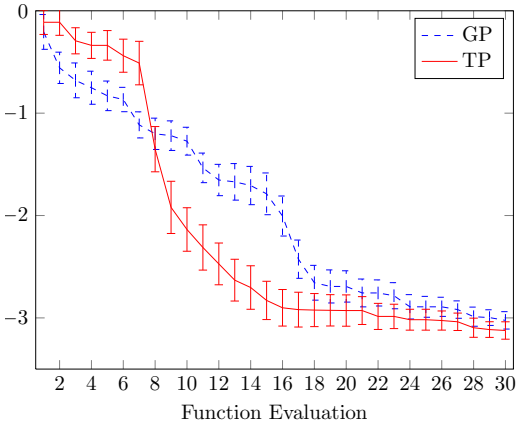
Other application areas where the TP should be successful are active and reinforcement learning, where having good predictive uncertainty can be crucial to guiding the search in an efficient way.



(a) Synthetic



(b) Branin-Hoo



(c) Hartmann

Fig. 2.7 Function evaluations for the (a) synthetic function, (b) Branin-Hoo function and (c) the Hartmann function. Evaluations under a Student-t process prior (solid red line) and a Gaussian process prior (dashed blue line) are shown. Error bars represent the standard deviation of 50 runs. In each panel we are minimising an objective function. The vertical axis represents the running minimum function value.

Chapter 3

Parallel Predictive Entropy Search for Batch Optimisation

Bayesian optimisation has been introduced as the task of finding the global optimum of a function, by sequentially deciding where to evaluate next, based on previous observations. However, in practice, it is often possible to evaluate a function at *multiple* points in *parallel*. For example, you may possess several identical robots on which you can test different gait parameters in parallel. Or your computer may have multiple cores on which you can run algorithms in parallel with different hyperparameter settings.

This chapter outlines a novel method for deciding on a *batch* of points where the function should be evaluated next in parallel. Most of the work is based on a collaboration with Zoubin Ghahramani, and has been published at NIPS [Shah and Ghahramani, 2015]. The key contribution we made, was to define a way to directly measure the quality of a set of points to evaluate the function in parallel, as opposed to previous methods which greedily choose one point at a time until a batch is filled. Our approach was to choose a set of points, which in expectation, maximally reduced our uncertainty about the location of the optimiser of the objective function. More concretely, we utilise the concept of *predictive entropy* to build our method upon.

3.1 Introduction

In this section, we introduce the concept of statistical entropy and how it may be used for Bayesian optimisation.

3.1.1 Entropy

In his seminal work on information theory, Shannon considered a way to describe how much information is gained by observing an outcome of a random variable [Shannon and Weaver, 1949]. More concretely, consider a discrete random variable, X, which takes values in $\{1, ..., n\}$ such that $\mathbb{P}[X = i] = p_i$. Define an information function, I, such that $\mathrm{I}[p_i]$ is the information acquired by observing the event $\{X = i\}$, which has probability p_i of occurring.

There are several properties that are intuitively desirable for an information function. Information should be non-negative since observing an event should never reduce how much information you have. No information is gained from observing an almost sure event (an event which occurs with probability 1). The higher the probability of an event, the less information observing that event gives you, therefore the information function should be monotonically decreasing. Finally, the total information from observing independent events should be equal to the sum of the information of observing each event. Formally, we require:

- $\mathrm{I}[p] \geq 0$,
- $\mathrm{I}[1] = 0$,
- $\mathrm{I}[p_1] > \mathrm{I}[p_2]$ when $p_1 < p_2$, and
- $\mathrm{I}[p_1 \times p_2] = \mathrm{I}[p_1] + \mathrm{I}[p_2]$.

Shannon's insight was that the function $\mathrm{I}[p] = -\log(p)$ was an ideal choice to satisfy the desired properties. The base of the logarithm is a choice left to the user leading to different units of information: bits for base 2, trits for base 3, nats for base e. Shannon defined $\mathrm{H}[X]$ as the *entropy* of a

discrete random variable, X, the expected information gain from observing an outcome of random variable X, such that

$$\mathrm{H}[X] = \mathbb{E}\Big[\mathrm{I}[X]\Big] = -\sum_{i=1}^{n} p_i \log(p_i). \tag{3.1}$$

The term 'entropy' is in fact an old one, and was chosen due to the similarity between the form of (3.1) and the *Gibbs entropy* in statistical thermodynamics [Gibbs, 1878]. As examples, the entropy of the outcome of (i) a toss of a fair coin is 1 bit, (ii) the roll of a fair die is log(6) bits, and (iii) the toss of a double headed coin is 0 bits.

Shannon stated a generalisation of entropy for a random variable defined on a continuous measure space, by defining the information function to be the negative logarithm of the probability density function and taking the expected value of this quantity using Lebesgue integration.

However, it can be shown that the limit of the discrete entropy as the number of discrete values a random variable may take tends to infinity, does not necessarily converge to this continuous definition of entropy [Marsh, 2013]. Nevertheless, the suggestion made by Shannon does have most of the desired properties of measuring the average information gained from an observation of a continuous random variable, and is commonly referred to as *differential entropy*. Let X now be a continuous random variable with density p_X and support $\mathcal{X}$, then the differential entropy of X is

$$\mathrm{H}[X] = \mathbb{E}\Big[\mathrm{I}[X]\Big] = -\int_{\mathcal{X}} p_X(x) \log\Big(p_X(x)\Big) dx. \tag{3.2}$$

Differential entropy may in fact be negative and is not invariant to a change in variables, so should be used with dimensionless variables.

3.1.2 Predictive Entropy Search

In this section we review work by Hennig and Schuler [2012] and Hernández-Lobato et al. [2014] which use the concept of entropy for Bayesian optimisation.

In the introduction we outline three strategies for deciding on where to evaluate a function next in a Bayesian optimisation framework, namely, expected improvement, probability of improvement and upper confidence bound. The former and latter operate in the units in which the unknown function is defined, whilst probability of improvement is defined in probability space. We proceed by defining an information theoretic acquisition function, measured in nats.

Let $\boldsymbol{x}^* \equiv \text{argmax}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$. Given a current set of observed function input and output pairs, $\mathcal{D}$, our posterior belief about $\boldsymbol{x}^*$ is summarised by $p(\boldsymbol{x}^*|\mathcal{D})$. One approach to the Bayesian optimisation problem would be to choose to evaluate in a location which, in expectation, maximally increases the information you gain about $\boldsymbol{x}^*$. Taking the negative differential entropy as a measure of information, the expected gain in information from making an observation y at location $\boldsymbol{x}$ is given by

$$\alpha(\boldsymbol{x}|\mathcal{D}) = \mathrm{H}[\boldsymbol{x}^*|\mathcal{D}] - \mathbb{E}_{p(y|\mathcal{D},\boldsymbol{x})}\Big[\mathrm{H}[\boldsymbol{x}^*|\mathcal{D}\cup\{\boldsymbol{x},y\}]\Big]. \tag{3.3}$$

Note that $\alpha(\boldsymbol{x}|\mathcal{D}) \geq 0$ for all $\boldsymbol{x} \in \mathcal{X}$, despite the fact that differential entropy may be negative, since $\mathrm{H}[\boldsymbol{x}^*|\mathcal{D}\cup\{\boldsymbol{x},y\}] \leq \mathrm{H}[\boldsymbol{x}^*|\mathcal{D}]$ for all $\boldsymbol{x} \in \mathcal{X}$, $y \in \mathbb{R}$ i.e. an extra observation cannot decrease your information about an unknown quantity. The density $p(\boldsymbol{x}^*|\mathcal{D})$ is a difficult one to work with as it is analytically intractable and only accessible via the posterior distribution $p(f|\mathcal{D})$.

It was the insight of Houlsby et al. [2012] to rearrange α into a more

computationally and analytically tractable quantity as follows

$$\begin{aligned}
\alpha(\boldsymbol{x}|\mathcal{D}) &= \mathrm{H}[\boldsymbol{x}^*|\mathcal{D}] - \mathbb{E}_{p(y|\mathcal{D},\boldsymbol{x})}\Big[\mathrm{H}[\boldsymbol{x}^*|\mathcal{D}\cup\{\boldsymbol{x},y\}]\Big] \\
&= -\int d\boldsymbol{x}^* p(\boldsymbol{x}^*|\mathcal{D}) \log\Big(p(\boldsymbol{x}^*|\mathcal{D})\Big) \\
&\quad + \int dy p(y|\mathcal{D},\boldsymbol{x}) \int d\boldsymbol{x}^* p(\boldsymbol{x}^*|\mathcal{D}\cup\{\boldsymbol{x},y\}) \log\Big(p(\boldsymbol{x}^*|\mathcal{D}\cup\{\boldsymbol{x},y\})\Big) \\
&= \int d\boldsymbol{x}^* \int dy p(\boldsymbol{x}^*,y|\mathcal{D},\boldsymbol{x}) \Bigg[\log\Big(p(\boldsymbol{x}^*|\mathcal{D}\cup\{\boldsymbol{x},y\})\Big) - \log\Big(p(\boldsymbol{x}^*|\mathcal{D})\Big)\Bigg] \\
&= \int d\boldsymbol{x}^* \int dy p(\boldsymbol{x}^*,y|\mathcal{D},\boldsymbol{x}) \Bigg[\log\Big(p(y|\mathcal{D},\boldsymbol{x},\boldsymbol{x}^*)\Big) - \log\Big(p(y|\mathcal{D},\boldsymbol{x})\Big)\Bigg] \\
&= -\int dy p(y|\mathcal{D},\boldsymbol{x}) \log\Big(p(y|\mathcal{D},\boldsymbol{x})\Big) \\
&\quad + \int d\boldsymbol{x}^* p(\boldsymbol{x}^*|\mathcal{D}) \int dy p(y|\mathcal{D},\boldsymbol{x},\boldsymbol{x}^*) \log\Big(p(y|\mathcal{D},\boldsymbol{x},\boldsymbol{x}^*)\Big) \\
&= \mathrm{H}[p(y|\mathcal{D},\boldsymbol{x})] - \mathbb{E}_{p(\boldsymbol{x}^*|\mathcal{D})}\Big[\mathrm{H}[y|\mathcal{D},\boldsymbol{x},\boldsymbol{x}^*]\Big].
\end{aligned}$$

The equivalence can be seen by noting that (3.3) is the conditional mutual information between y and $\boldsymbol{x}^*$ given $\mathcal{D}$ and $\boldsymbol{x}$, and that the conditional mutual information is a symmetric function [MacKay, 1992]. The quantity α has been transformed from a quantity involving Lebesgue integrals with respect to measures over $\boldsymbol{x}^*$, the function optimiser, to a quantity involving Lebesgue integrals with respect to measures over y, the function value. Measures over the function value are computationally tractable under the assumption of a Gaussian process model over f, making the new representation of α easier to work with. Hernández-Lobato et al. [2014] exploit this property to derive simple approximations and an algorithm based on expectation propagation [Minka, 2001] to compute and optimise α.

3.2 Parallel Predictive Entropy Search (PPES)

In this section we describe our contribution to extending the method of predictive entropy search to the setting of batch Bayesian optimisation, where multiple function locations may be probed in parallel.

3.2.1 Problem Statement and Setup

More formally, our aim is maximise an objective function $f : \mathcal{X} \to \mathbb{R}$, which is unknown but can be (noisily) evaluated pointwise at multiple locations in parallel. We assume $\mathcal{X}$ is a compact subset of $\mathbb{R}^D$. At each decision, we must select a set of Q points $\mathcal{S}_t = \{\boldsymbol{x}_{t,1}, ..., \boldsymbol{x}_{t,Q}\} \subset \mathcal{X}$, where the objective function would next be evaluated in parallel. Each evaluation leads to a scalar observation $y_{t,q} = f(\boldsymbol{x}_{t,q}) + \epsilon_{t,q}$, where we assume $\epsilon_{t,q} \sim \mathcal{N}(0, \sigma^2)$ i.i.d. We wish to minimise a future simple *regret*, $r_T = [f(\boldsymbol{x}^*) - f(\tilde{\boldsymbol{x}}_T)]$, where $\boldsymbol{x}^* \in \text{argmax}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ is an optimal decision (assumed to exist) and $\tilde{\boldsymbol{x}}_T$ is our guess of where the maximiser of f is after evaluating T batches of input points. It is intractable to make decisions T steps ahead in the setting described, therefore it is common to consider the regret of the very next decision. In this work, we shall assume f is a draw from a Gaussian process with constant mean $\lambda \in \mathbb{R}$ and differentiable kernel function $k : \mathcal{X}^2 \to \mathbb{R}$.

Our approach is to maximise information [MacKay, 1992] about the location of the global maximiser $\boldsymbol{x}^*$, which we measure in terms of the negative differential entropy of $p(\boldsymbol{x}^*|\mathcal{D})$. Analogous to Hernández-Lobato et al. [2014], PPES aims to choose the set of Q points, $\mathcal{S}_t = \{\boldsymbol{x}_q\}_{q=1}^Q$, which maximises

$$a_{\text{PPES}}(\mathcal{S}_t|\mathcal{D}) = \text{H}\Big[\boldsymbol{x}^*|\mathcal{D}\Big] - \mathbb{E}_{p\Big(\{y_q\}_{q=1}^Q \Big| \mathcal{D}, \mathcal{S}_t\Big)}\Big[\text{H}\Big[\boldsymbol{x}^*|\mathcal{D} \cup \{\boldsymbol{x}_q, y_q\}_{q=1}^Q\Big]\Big]. \quad (3.4)$$

Evaluating (3.4) exactly is typically infeasible. The prohibitive aspects are that $p\Big(\boldsymbol{x}^*|\mathcal{D} \cup \{\boldsymbol{x}_q, y_q\}_{q=1}^Q\Big)$ would have to be evaluated for many different combinations of $\{\boldsymbol{x}_q, y_q\}_{q=1}^Q$, and the entropy computations are not analytically tractable in themselves. Significant approximations need to be made to (3.4) before it becomes practically useful [Hennig and Schuler, 2012]. A convenient equivalent formulation of a_{PPES} can be written as the mutual information between $\boldsymbol{x}^*$ and $\{y_q\}_{q=1}^Q$ given $\mathcal{D}$ [Houlsby et al., 2012]. By symmetry of the mutual information, we can rewrite a_{PPES} as

$$a_{\text{PPES}}(\mathcal{S}_t|\mathcal{D}) = \text{H}\Big[\{y_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t\Big] - \mathbb{E}_{p(\boldsymbol{x}^*|\mathcal{D})}\Big[\text{H}\Big[\{y_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big]\Big], \quad (3.5)$$

where $p\Big(\{y_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big)$ is the joint posterior predictive distribution for $\{y_q\}_{q=1}^Q$ given the observed data, $\mathcal{D}$ and the location of the global maximiser of f. The key advantage of the formulation in (3.5), is that the objective is based on entropies of predictive distributions of the observations, which are much more easily approximated than the entropies of distributions on $\boldsymbol{x}^*$.

In fact, the first term of (3.5) can be computed analytically. For convenience, $f_q \equiv f(\boldsymbol{x}_q)$. Since $\{f_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t$ is multivariate Gaussian with covariance $\mathbf{K}$, then

$$\mathrm{H}\Big[\{y_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t\Big] = 0.5 \log[\det(2\pi e(\mathbf{K} + \sigma^2\mathbf{I}))]. \tag{3.6}$$

We develop an approach to approximate the expectation of the conditional predictive entropy in (3.5), using an expectation propagation based method which we discuss in the following subsection.

3.2.2 How to Approximate the Conditional Predictive Entropy

Assuming a sample of $\boldsymbol{x}^*$, we discuss our approach to approximating $\mathrm{H}\Big[\{y_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big]$ in (3.5) for a set of query Q points, $\mathcal{S}_t$. Note that we can write

$$p\Big(\{y_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big) = \int p\Big(\{f_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big) \prod_{q=1}^Q p(y_q|f_q)\ df_1 ... df_Q, \tag{3.7}$$

where $p\Big(\{f_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big)$ is the posterior distribution of the objective function at the locations $\boldsymbol{x}_q \in \mathcal{S}_t$, given previous evaluations $\mathcal{D}$, and that $\boldsymbol{x}^*$ is the global maximiser of f. Recall that $p(y_q|f_q)$ is Gaussian for each q. Our approach will be to derive a Gaussian approximation to $p\Big(\{f_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big)$, which would lead to an analytic approximation to the integral in (3.7).

The posterior predictive distribution of the Gaussian process, $p\Big(\{f_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t\Big)$, is multivariate Gaussian distributed. However, by further conditioning on the location $\boldsymbol{x}^*$, the global maximiser of f, we impose the condition that $f(\boldsymbol{x}) \leq f(\boldsymbol{x}^\star)$ for any $\boldsymbol{x} \in \mathcal{X}$. Imposing this constraint for all $\boldsymbol{x} \in \mathcal{X}$ is

extremely difficult and makes the computation of $p\Big(\{f_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big)$ highly intractable. We instead impose the following two conditions:

(i) $f(\boldsymbol{x}) \leq f(\boldsymbol{x}^\star)$ for each $\boldsymbol{x} \in \mathcal{S}_t$, and

(ii) $f(\boldsymbol{x}^\star) \geq y_{\max} + \epsilon$, where $y_{\max}$ is the largest observed noisy objective function value and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. In fact, we marginalise ϵ out:

$$\int \mathbb{I}(f(\boldsymbol{x}^\star) \geq y_{\max} + \epsilon)\mathcal{N}(\epsilon; 0, \sigma^2)d\epsilon = \Phi\Big(\frac{f^\star - y_{\max}}{\sigma}\Big).$$

Constraint (i) is equivalent to imposing that $f(\boldsymbol{x}^\star)$ is larger than each other objective function value at current query locations. Condition (ii) makes $f(\boldsymbol{x}^\star)$ larger than all previous objective function evaluations, accounting for noise. Denoting the two conditions $\mathfrak{C}$, and the variables $\boldsymbol{f} \equiv [f_1, ..., f_Q]^\top$ and $\boldsymbol{f}_+ \equiv [\boldsymbol{f}; f^\star]$, where $f^\star \equiv f(\boldsymbol{x}^*)$, we incorporate the conditions as follows

$$p\Big(\boldsymbol{f}|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big) \approx \int p\Big(\boldsymbol{f}_+|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big)\Phi\Big(\frac{f^\star - y_{\max}}{\sigma}\Big)\prod_{q=1}^{Q}\mathbb{I}(f^\star \geq f_q)\, df^\star, \quad (3.8)$$

where $\mathbb{I}(.)$ is an indicator function, returning 1 when its argument is true and 0 otherwise. The integral in (3.8) can be approximated using expectation propagation [Minka, 2001].

3.2.3 Expectation Propagation Inference

The Gaussian process predictive $p(\boldsymbol{f}_+|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*)$ is $\mathcal{N}(\boldsymbol{f}_+; \mathbf{m}_+, \mathbf{K}_+)$. We approximate each $\mathbb{I}(f^\star \geq f_q)$ in the integrand of (3.8), with a function of the form $\tilde{Z}_q\mathcal{N}(\boldsymbol{c}_q^\top \boldsymbol{f}_+; \tilde{\mu}_q, \tilde{\tau}_q)$, where $\boldsymbol{c}_q$ is a vector of length $Q+1$ with q^{th} entry -1, $Q+1^{\text{st}}$ entry 1, and remaining entries 0. Similarly we approximate $\Phi\Big(\frac{f^\star - y_{\max}}{\sigma}\Big)$ with $\tilde{Z}_{Q+1}\mathcal{N}(\boldsymbol{c}_{Q+1}^\top \boldsymbol{f}_+; \tilde{\mu}_{Q+1}, \tilde{\tau}_{Q+1})$, where $\boldsymbol{c}_{Q+1}$ is a vector of length $Q+1$ with final entry 1 and remaining entries 0. Each $\tilde{Z}_q$ and $\tilde{\tau}_q$ is positive, whilst each $\tilde{\mu}_q \in \mathbb{R}$. We have approximated each indicator function and Gaussian c.d.f. with a scaled Gaussian p.d.f. The *site parameters*, $\{\tilde{Z}_q, \tilde{\mu}_q, \tilde{\tau}_q\}_{q=1}^{Q+1}$ are to be optimised in a method we shall describe shortly.

The final approximate integrand of (3.8) is $w(\boldsymbol{f}_+)$, given by

$$w(\boldsymbol{f}_+) = \mathcal{N}(\boldsymbol{f}_+; \mathbf{m}_+, \mathbf{K}_+) \prod_{q=1}^{Q+1} \tilde{Z}_q \mathcal{N}(\boldsymbol{c}_q^\top \boldsymbol{f}_+; \tilde{\mu}_q, \tilde{\tau}_q). \tag{3.9}$$

Since products of Gaussian p.d.f.s lead to unnormalised Gaussian p.d.f.s, we find that $w(\boldsymbol{f}_+) = Z\mathcal{N}(\boldsymbol{f}_+; \boldsymbol{\mu}_+, \boldsymbol{\Sigma}_+)$, where

$$\boldsymbol{\mu}_+ = \boldsymbol{\Sigma}_+ \Big(\mathbf{K}_+^{-1} \mathbf{m}_+ + \sum_{q=1}^{Q+1} \frac{\tilde{\mu}_q}{\tilde{\tau}_q} \boldsymbol{c}_q \boldsymbol{c}_q^\top \Big)^{-1}, \tag{3.10}$$

$$\boldsymbol{\Sigma}_+ = \left(\mathbf{K}_+^{-1} + \sum_{q=1}^{Q+1} \frac{1}{\tilde{\tau}_q} \boldsymbol{c}_q \boldsymbol{c}_q^\top \right)^{-1}, \tag{3.11}$$

$$\begin{aligned} \log Z = &-\frac{1}{2}\Big(\mathbf{m}_+ \mathbf{K}_+^{-1} \mathbf{m}_+ + \log |\mathbf{K}_+| \Big) \\ &+ \sum_{q=1}^{Q} \left(\log \tilde{Z}_q - \frac{1}{2}\Big(\frac{\tilde{\mu}_q^2}{\tilde{\tau}_q} + \log \tilde{\sigma}_q^2 + \log(2\pi) \Big) \right) \\ &+ \frac{1}{2}\Big(\boldsymbol{\mu}_+ \boldsymbol{\Sigma}_+^{-1} \boldsymbol{\mu}_+ + \log |\boldsymbol{\Sigma}_+| \Big). \end{aligned} \tag{3.12}$$

We now describe the steps required to update the site parameters. We closely follow the derivations in Cunningham et al. [2013]. We first compute the *cavity* distributions,

$$w^{\backslash q}(\boldsymbol{f}_+) = \frac{w(\boldsymbol{f}_+)}{\tilde{Z}_q \mathcal{N}(\boldsymbol{c}_q^\top \boldsymbol{f}_+; \tilde{\mu}_q, \tilde{\tau}_q)} \tag{3.13}$$

and compute their Gaussian parameters. Since we are dividing a Gaussian p.d.f. by another Gaussian p.d.f. we have simple parameter updates given by

$$\tau_{\backslash q} = \Big((\boldsymbol{c}_q^\top \boldsymbol{\Sigma}_+^{-1} \boldsymbol{c}_q)^{-1} - \tilde{\tau}_q^{-1} \Big)^{-1} \tag{3.14}$$

$$\mu_{\backslash q} = \tau_{\backslash q} \left(\frac{\boldsymbol{c}_q^\top \boldsymbol{\mu}_+}{\boldsymbol{c}_q^\top \boldsymbol{\Sigma}_+ \boldsymbol{c}_q} - \frac{\tilde{\mu}_q}{\tilde{\tau}_q} \right). \tag{3.15}$$

The next step of EP is the *projection* step and requires moment matching $\tilde{Z}_q \mathcal{N}(\boldsymbol{c}_q^\top \boldsymbol{f}_+; \tilde{\mu}_q, \tilde{\tau}_q) w^{\backslash q}(\boldsymbol{f}_+)$ with $t_q(\boldsymbol{f}_+) w^{\backslash q}(\boldsymbol{f}_+)$, where $t_q(\boldsymbol{f}_+)$ is the true q^{th} factor being approximated. Therefore, $t_q(\boldsymbol{f}_+) \equiv \mathbb{I}[\boldsymbol{c}_q^\top \boldsymbol{f}_+ \geq 0]$ for $1 \leq q \leq Q$

and $t_{Q+1}(\boldsymbol{f}_+) \equiv \Phi\Big(\frac{\boldsymbol{c}_{Q+1}^\top \boldsymbol{f}_+ - y_{\max}}{\sigma}\Big)$. We use derivatives of the logarithm of the zeroth moment [Minka, 2001] to compute the parameters

$$\begin{aligned}
\hat{Z}_q &= \int t_q(\boldsymbol{f}_+) w^{\backslash q}(\boldsymbol{f}_+) d\boldsymbol{f}_+ \\
&= \Phi(\beta_q), && (3.16)\\
\hat{\mu}_q &= \mu_{\backslash q} + \tau_{\backslash q} \frac{\partial \log \hat{Z}_q}{\partial \mu_{\backslash q}} \\
&= \mu_{\backslash q} + \sqrt{\tau_{\backslash q}} \frac{\phi(\beta_q)}{\Phi(\beta_q)}, && (3.17)\\
\hat{\tau}_q &= \tau_{\backslash q} - \tau_{\backslash q}^2 \Big(\Big(\frac{\partial \log \hat{Z}_q}{\partial \mu_{\backslash q}} \Big)^2 - 2 \frac{\partial \log \hat{Z}_q}{\partial \tau_{\backslash q}} \Big) \\
&= \tau_{\backslash q} - \frac{\tau_{\backslash q}^3}{\mu_{\backslash q}^2} \Big(\frac{\phi(\beta_q)}{\Phi(\beta_q)} \Big) \Big(\frac{\phi(\beta_q)}{\Phi(\beta_q)} + \beta_q \Big), && (3.18)
\end{aligned}$$

where $\beta_q = \frac{\mu_{\backslash q}}{\sqrt{\tau_{\backslash q}}}$ for $q \leq Q$ and $\beta_{Q+1} = \Phi\Big(\frac{\mu_{\backslash q} - y_{\max}}{\sqrt{\sigma^2 + \tau_{\backslash q}}}\Big)$. To complete the projection step, we update the site parameters to achieve the moments computed above by setting

$$\tilde{\tau}_q = \Big(\hat{\tau}_q^{-1} - \tau_{\backslash q}^{-1} \Big)^{-1}, \tag{3.19}$$

$$\tilde{\mu}_q = \tilde{\tau}_q \Big(\hat{\tau}_q^{-1} \hat{\mu}_q - \tau_{\backslash q}^{-1} \mu_{\backslash q} \Big)^{-1}. \tag{3.20}$$

$$\tilde{Z}_q = \hat{Z}_q \sqrt{2\pi} \sqrt{\tau_{\backslash q} + \tilde{\tau}_q} \exp\left[\frac{1}{2} \frac{\Big(\mu_{\backslash q} - \tilde{\mu}_q\Big)^2}{\Big(\tau_{\backslash q} + \tilde{\tau}_q\Big)} \right]. \tag{3.21}$$

Finally we update the parameters $\boldsymbol{\mu}_+$ and $\boldsymbol{\Sigma}_+$ as in equations 3.10 and 3.11, and repeat the process until convergence.

3.2.4 The PPES Approximation

Once the site parameters $\{\tilde{Z}_q, \tilde{\mu}_q, \tilde{\tau}_q\}_{q=1}^{Q+1}$ have been learnt through expectation propagation, we have $p\Big(\boldsymbol{f}_+ | \mathcal{D}, \mathcal{S}_t, \mathcal{C}\Big) \approx \mathcal{N}(\boldsymbol{f}_+; \boldsymbol{\mu}_+, \boldsymbol{\Sigma}_+)$. Since multivariate Gaussians are consistent under marginalisation, a convenient corollary is that $p\Big(\boldsymbol{f} | \mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big) \approx \mathcal{N}(\boldsymbol{f}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu}$ is the vector containing the first Q elements of $\boldsymbol{\mu}_+$, and $\boldsymbol{\Sigma}$ is the matrix containing the first Q rows and

columns of $\boldsymbol{\Sigma}_+$. Since sums of independent Gaussians are also Gaussian distributed, we see that $p\Big(\{y_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big) \approx \mathcal{N}([y_1, ..., y_Q]^\top; \boldsymbol{\mu}, \boldsymbol{\Sigma} + \sigma^2\mathbf{I})$. The final convenient attribute of our Gaussian approximation, is that the differential entropy of a multivariate Gaussian can be computed analytically, such that $\mathrm{H}\Big[\{y_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big] \approx 0.5 \log[\det(2\pi e(\boldsymbol{\Sigma} + \sigma^2\mathbf{I}))]$.

3.2.5 Sampling from the Posterior over the Global Maximiser

So far, we have considered how to approximate $\mathrm{H}\Big[\{y_q\}_{q=1}^Q|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*\Big]$, given the global maximiser, $\boldsymbol{x}^*$. We in fact would like the expected value of this quantity over the posterior distribution of the global maximiser, $p(\boldsymbol{x}^*|\mathcal{D})$. Literally, $p(\boldsymbol{x}^*|\mathcal{D}) \equiv p(f(\boldsymbol{x}^*) = \max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})|\mathcal{D})$, the posterior probability that $\boldsymbol{x}^*$ is the global maximiser of f. Computing the distribution $p(\boldsymbol{x}^*|\mathcal{D})$ is intractable, but it is possible to approximately sample from it and compute a Monte Carlo based approximation of the desired expectation. We consider two approaches to sampling from the posterior of the global maximiser: (i) a maximum a posteriori (MAP) method, and (ii) a random feature approach.

MAP sample from $p(\boldsymbol{x}^*|\mathcal{D})$. The MAP of $p(\boldsymbol{x}^*|\mathcal{D})$ is its posterior mode, given by $\boldsymbol{x}^*_{\mathrm{MAP}} = \mathrm{argmax}_{\boldsymbol{x}^*\in\mathcal{X}}\, p(\boldsymbol{x}^*|\mathcal{D})$. We may approximate the expected value of the predictive entropy by replacing the posterior distribution of $\boldsymbol{x}^*$ with a single point estimate at $\boldsymbol{x}^*_{\mathrm{MAP}}$. There are two key advantages to using the MAP estimate in this way. Firstly, it is simple to compute $\boldsymbol{x}^*_{\mathrm{MAP}}$, as it is the global maximiser of the posterior mean of f given the observations $\mathcal{D}$. Secondly, choosing to use $\boldsymbol{x}^*_{\mathrm{MAP}}$ assists the EP algorithm developed above to converge as desired. This is because the condition $f(\boldsymbol{x}^*) \geq f(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}$ is easy to enforce when $\boldsymbol{x}^* = \boldsymbol{x}^*_{\mathrm{MAP}}$, the global maximiser of the posterior mean of f. When $\boldsymbol{x}^*$ is sampled such that the posterior mean at $\boldsymbol{x}^*$ is significantly suboptimal, the EP approximation may be poor. Whilst using the MAP estimate approximation is convenient, it is after all a point estimate and fails to characterise the full posterior distribution. We therefore consider a method to draw samples from $p(\boldsymbol{x}^*|\mathcal{D})$ using random features.

Random Feature Samples from $p(\boldsymbol{x}^*|\mathcal{D})$. A naive approach to sampling from $p(\boldsymbol{x}^*|\mathcal{D})$ would be to sample $g \sim p(f|\mathcal{D})$, and choosing $\text{argmax}_{\boldsymbol{x}\in\mathcal{X}} g$. Unfortunately, this would require sampling g over an uncountably infinite space, which is infeasible. A slightly less naive method would be to sequentially construct g, whilst optimising it, instead of evaluating it everywhere in $\mathcal{X}$. However, this approach would have cost $\mathcal{O}(m^3)$ where m is the number of function evaluations of g necessary to find its optimum. We propose as in Hernández-Lobato et al. [2014], to sample and optimise an analytic approximation to g.

By Bochner's theorem [Bochner, 1959], a stationary kernel function, k, has a Fourier dual $s(\boldsymbol{w})$, which is equal to the spectral density of k. Setting $p(\boldsymbol{w}) = s(\boldsymbol{w})/\alpha$, a normalised density, we can write

$$k(\boldsymbol{x}, \boldsymbol{x}') = \alpha \mathbb{E}_{p(\boldsymbol{w})}[e^{-i\boldsymbol{w}^\top(\boldsymbol{x}-\boldsymbol{x}')}] = 2\alpha \mathbb{E}_{p(\boldsymbol{w},b)}[\cos(\boldsymbol{w}^\top \boldsymbol{x} + b)\cos(\boldsymbol{w}^\top \boldsymbol{x}' + b)], \tag{3.22}$$

where $b \sim U[0, 2\pi]$. Let $\boldsymbol{\phi}(\boldsymbol{x}) = \sqrt{2\alpha/m}\cos(\mathbf{W}\boldsymbol{x} + \boldsymbol{b})$ denote an m-dimensional feature mapping where $\mathbf{W}$ and $\boldsymbol{b}$ consist of m stacked samples from $p(\boldsymbol{w}, b)$, then the kernel k can be approximated by the inner product of these features, $k(\boldsymbol{x}, \boldsymbol{x}') \approx \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}')$ [Rahimi and Recht, 2007]. The linear model $g(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\theta} + \lambda$ where $\boldsymbol{\theta}|\mathcal{D} \sim \mathcal{N}(\mathbf{A}^{-1}\boldsymbol{\Phi}^\top(\boldsymbol{y} - \lambda\mathbf{1}), \sigma^2\mathbf{A}^{-1})$ is an approximate sample from $p(f|\mathcal{D})$, where $\boldsymbol{y}$ is a vector of objective function evaluations, $\mathbf{A} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma^2\mathbf{I}$ and $\boldsymbol{\Phi}^\top = [\boldsymbol{\phi}(\boldsymbol{x}_1)...\boldsymbol{\phi}(\boldsymbol{x}_n)]$. In fact, $\lim_{m\to\infty} g$ is a true sample from $p(f|\mathcal{D})$ [Neal, 1995].

The generative process above suggests the following approach to approximately sampling from $p(\boldsymbol{x}^*|\mathcal{D})$: (i) sample random features $\boldsymbol{\phi}^{(i)}$ and corresponding posterior weights $\boldsymbol{\theta}^{(i)}$ using the process above, (ii) construct $g^{(i)}(\boldsymbol{x}) = \boldsymbol{\phi}^{(i)}(\boldsymbol{x})^\top\boldsymbol{\theta}^{(i)} + \lambda$, and (iii) finally compute $\boldsymbol{x}^{\star(i)} = \text{argmax}_{\boldsymbol{x}\in\mathcal{X}}\, g^{(i)}(\boldsymbol{x})$ using gradient based methods.

3.2.6 Optimising the PPES Approximation

Let $\boldsymbol{\psi}$ denote the set of kernel parameters and the observation noise variance, σ^2. Our posterior belief about $\boldsymbol{\psi}$ is summarised by the posterior distribution

$p(\boldsymbol{\psi}|\mathcal{D}) \propto p(\boldsymbol{\psi})p(\mathcal{D}|\boldsymbol{\psi})$, where $p(\boldsymbol{\psi})$ is our prior belief about $\boldsymbol{\psi}$ and $p(\mathcal{D}|\boldsymbol{\psi})$ is the GP marginal likelihood of the data given the parameters $\boldsymbol{\psi}$. For a fully Bayesian treatment of $\boldsymbol{\psi}$, we must marginalise a_{PPES} with respect to $p(\boldsymbol{\psi}|\mathcal{D})$. The expectation with respect to the posterior distribution of $\boldsymbol{\psi}$ is approximated with Monte Carlo samples (see equation (3.23)). A similar approach is taken in Snoek et al. [2012] and Hernández-Lobato et al. [2014]. Combining the EP based method to approximate the predictive entropy with either of the two methods discussed previously to approximately sample from $p(\boldsymbol{x}^*|\mathcal{D})$, we can construct $\hat{a}_{\text{PPES}}$ an approximation to (3.5), defined by

$$\hat{a}_{\text{PPES}}(\mathcal{S}_t|\mathcal{D}) = \frac{1}{2M}\sum_{i=1}^{M}\Big[\log[\det(\mathbf{K}^{(i)} + \sigma^{2(i)}\mathbf{I})] - \log[\det(\boldsymbol{\Sigma}^{(i)} + \sigma^{2(i)}\mathbf{I})]\Big], \tag{3.23}$$

where $\mathbf{K}^{(i)}$ is constructed using $\boldsymbol{\psi}^{(i)}$ the i^{th} sample of M from $p(\boldsymbol{\psi}|\mathcal{D})$, $\boldsymbol{\Sigma}^{(i)}$ is constructed as above, assuming the global maximiser is $\boldsymbol{x}^{*(i)} \sim p(\boldsymbol{x}^*|\mathcal{D}, \boldsymbol{\psi}^{(i)})$. The PPES approximation is simple and amenable to gradient based optimisation. Our goal is to choose $\mathcal{S}_t = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_Q\}$ which maximises $\hat{a}_{\text{PPES}}$ in (3.23). Since our kernel function is differentiable, we may consider taking the derivative of $\hat{a}_{\text{PPES}}$ with respect to $x_{q,d}$, the d^{th} component of $\boldsymbol{x}_q$,

$$\frac{\partial \hat{a}_{\text{PPES}}}{\partial x_{q,d}} = \frac{1}{2M}\sum_{i=1}^{M}\left[\text{Tr}\left[(\mathbf{K}^{(i)}+\sigma^{2(i)}\mathbf{I})^{-1}\frac{\partial \mathbf{K}^{(i)}}{\partial x_{q,d}}\right] - \text{Tr}\left[(\boldsymbol{\Sigma}^{(i)}+\sigma^{2(i)}\mathbf{I})^{-1}\frac{\partial \boldsymbol{\Sigma}^{(i)}}{\partial x_{q,d}}\right]\right]. \tag{3.24}$$

Computing $\frac{\partial \mathbf{K}^{(i)}}{\partial x_{q,d}}$ is simple directly from the definition of the chosen kernel function. $\boldsymbol{\Sigma}^{(i)}$ is a function of $\mathbf{K}^{(i)}$, $\{\boldsymbol{c}_q\}_{q=1}^{Q+1}$ and $\{\tilde{\sigma}_q^{(i)}\}_{q=1}^{Q+1}$, and we know how to compute $\frac{\partial \mathbf{K}^{(i)}}{\partial x_{q,d}}$, and that each $\boldsymbol{c}_q$ is a constant vector. Hence our only concern is how the EP site parameters, $\{\tilde{\sigma}_q^{(i)}\}_{q=1}^{Q+1}$, vary with $x_{q,d}$. Rather remarkably, we may invoke a result from Section 2.1 of Seeger [2008], which says that converged site parameters, $\{\tilde{Z}_q, \tilde{\mu}_q, \tilde{\sigma}_q\}_{q=1}^{Q+1}$, have 0 derivative with respect to parameters of $p(\boldsymbol{f}_+|\mathcal{D}, \mathcal{S}_t, \boldsymbol{x}^*)$. There is a key distinction between *explicit* dependencies (where $\boldsymbol{\Sigma}$ actually depends on $\mathbf{K}$) and *implicit* dependencies where a site parameter, $\tilde{\sigma}_q$, might depend implicitly on $\mathbf{K}$. A similar approach is taken in Cunningham et al. [2013], and discussed in

Rasmussen and Williams [2006]. We therefore compute

$$\frac{\partial \boldsymbol{\Sigma}_+^{(i)}}{\partial x_{q,d}} = \boldsymbol{\Sigma}_+^{(i)} \mathbf{K}_+^{(i)-1} \frac{\partial \mathbf{K}_+^{(i)}}{\partial x_{q,d}} \mathbf{K}_+^{(i)-1} \boldsymbol{\Sigma}_+^{(i)}. \tag{3.25}$$

On first inspection, it may seem computationally too expensive to compute derivatives with respect to each q and d. However, note that we may compute and store the matrices $\mathbf{K}_+^{(i)-1}\boldsymbol{\Sigma}_+^{(i)}$, $(\mathbf{K}^{(i)} + \sigma^{2(i)}\mathbf{I})^{-1}$ and $(\boldsymbol{\Sigma}^{(i)} + \sigma^{2(i)}\mathbf{I})^{-1}$ once, and that $\frac{\partial \mathbf{K}_+^{(i)}}{\partial x_{q,d}}$ is symmetric with exactly one non-zero row and non-zero column, which can be exploited for fast matrix multiplication and trace computations.

Note that the approximation to the acquisition function we have developed, $\hat{a}_{\text{PPES}}$, is defined and optimised on the entire set of points $\mathcal{S}_t$. This results in a non-greedy batch selection procedure, where the Q points are decided upon in one go. This was not the case for any other method proposed in the literature until our NIPS 2015 submission to the best of our knowledge.

3.3 Related Work

Most Bayesian optimisation research focuses on choosing a single point to query at each decision i.e. $Q = 1$. A popular strategy in this setting is to choose the point with highest expected improvement over the current best evaluation, i.e. the maximiser of

$$a_{\text{EI}}(\boldsymbol{x}|\mathcal{D}) = \mathbb{E}\Big[\max(f(\boldsymbol{x}) - f(\boldsymbol{x}_{\text{best}}), 0)\Big|\mathcal{D}\Big] = \sigma(\boldsymbol{x})\Big[\phi\Big(\tau(\boldsymbol{x})\Big) + \tau(\boldsymbol{x})\Phi\Big(\tau(\boldsymbol{x})\Big)\Big],$$

where $\mathcal{D}$ is the set of observations, $\boldsymbol{x}_{\text{best}}$ is the best evaluation point so far, $\sigma(\boldsymbol{x}) = \sqrt{\text{Var}[f(\boldsymbol{x})|\mathcal{D}]}$, $\mu(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})|\mathcal{D}]$, $\tau(\boldsymbol{x}) = (\mu(\boldsymbol{x}) - f(\boldsymbol{x}_{\text{best}}))/\sigma(\boldsymbol{x})$ and $\phi(.)$ and $\Phi(.)$ are the standard Gaussian p.d.f. and c.d.f.

Aside from being an intuitive approach, a key advantage of using the *expected improvement* strategy is in the fact that it is computable analytically and is differentiable if the kernel, k is differentiable. This makes the problem of

finding $\text{argmax}_{\boldsymbol{x}\in\mathcal{X}} a_{\text{EI}}(\boldsymbol{x}|\mathcal{D})$ amenable to a plethora of gradient based optimisation methods. Unfortunately, the corresponding strategy for selecting $Q > 1$ points to evaluate in parallel does not lead to an analytic expression. Ginsbourger et al. [2011] considered an approach which sequentially used the EI criterion to greedily choose a batch of points to query next. Snoek et al. [2012] formalised and utilised this approach by defining

$$a_{\text{EI-MC}}\Big(\boldsymbol{x}|\mathcal{D}, \{\boldsymbol{x}_{q'}\}_{q'=1}^{q}\Big) = \int_{\mathcal{X}^q} a_{\text{EI}}\Big(\boldsymbol{x}|\mathcal{D} \cup \{\boldsymbol{x}_{q'}, y_{q'}\}_{q'=1}^{q}\Big) \times p\Big(\{y_{q'}\}_{q'=1}^{q}|\mathcal{D}, \{\boldsymbol{x}_{q'}\}_{q'=1}^{q}\Big) dy_1..dy_q,$$

the expected gain in evaluating $\boldsymbol{x}$ after evaluating $\{\boldsymbol{x}_{q'}, y_{q'}\}_{q'=1}^{q}$, which can be approximated using Monte Carlo samples, hence the name EI-MC. Choosing a batch of points $\mathcal{S}_t$ using the EI-MC policy is *doubly greedy*: (i) the EI criterion is greedy as it inherently aims to minimise one-step regret, r_t, and (ii) the EI-MC approach starts with an empty set and populates it sequentially (and hence greedily), deciding the best single point to include until $|\mathcal{S}_t| = Q$.

A similar but different approach called *simulated matching* (SM) was introduced by Azimi et al. [2010]. Let π be a baseline policy which chooses a single point to evaluate next (e.g. EI). SM aims to select a batch $\mathcal{S}_t$ of size Q, which includes a point 'close to' the best point which π would have chosen when applied sequentially Q times, with high probability. Formally, SM aims to maximise

$$a_{\text{SM}}(\mathcal{S}_t|\mathcal{D}) = -\mathbb{E}_{S_\pi^Q}\left[\mathbb{E}_f\left[\min_{\boldsymbol{x}\in\mathcal{S}_t}(\boldsymbol{x} - \text{argmax}_{\boldsymbol{x'}\in S_\pi^Q} f(\boldsymbol{x'}))^2 \middle| \mathcal{D}, S_\pi^Q\right]\right],$$

where S_π^Q is the set of Q points which policy π would query if employed sequentially. A greedy k-medoids based algorithm is proposed to approximately maximise the objective, which the authors justify by the submodularity of the objective function.

The *upper confidence bound* (UCB) strategy [Srinivas et al., 2010] is another method used by practitioners to decide where to evaluate an objective

function next. The UCB approach is to maximise

$$a_{\mathrm{UCB}}(\boldsymbol{x}|\mathcal{D}) = \mu(\boldsymbol{x}) + \alpha_t^{1/2}\sigma(\boldsymbol{x}),$$

where α_t is a domain-specific time-varying positive parameter which trades off exploration and exploitation. In order to extend this approach to the parallel setting, Desautels et al. [2012] noted that the predictive variance of a Gaussian process depends only on where observations are made, and not the observations themselves. Therefore, they suggested the GP-BUCB method, which greedily populates the set $\mathcal{S}_t$ by maximising a UCB type equation Q times sequentially, updating σ at each step, whilst maintaining the same μ for each batch. This approach has the benefit that it is fairly simple and intuitive to implement, however, it makes the very strong assumption that observations all take values of the GP predictive mean, which can be thought of as a maximum a posteriori approximation. This MAP approach may potentially dramatically reduce the benefits of being fully Bayesian.

Finally, a variant of the GP-UCB was proposed by Contal et al. [2013]. The first point of the set $\mathcal{S}_t$ is chosen by optimising the UCB objective. Thereafter, a 'relevant region' $\mathcal{R}_t \subset \mathcal{X}$ which contains the maximiser of f with high probability is defined. Points are greedily chosen from this region to maximise the information gain about f, measured by expected reduction in entropy, until $|\mathcal{S}_t| = Q$. This method was named Gaussian process upper confidence bound with pure exploration (GP-UCB-PE).

Each approach discussed here resorts to a greedy batch selection process. To the best of our knowledge, no batch Bayesian optimisation method to date has avoided a greedy algorithm. We avoid a greedy batch selection approach with PPES. The benefits of our entropy based, non-greedy approach are explored in the next section.

3.4 Empirical Study

In this section, we study the performance of PPES in comparison to aforementioned methods. We model f as a Gaussian process with constant mean

λ and covariance kernel k. Observations of the objective function are considered to be independently drawn from $\mathcal{N}(f(\boldsymbol{x}), \sigma^2)$. In our experiments, we choose to use a squared-exponential kernel of the form

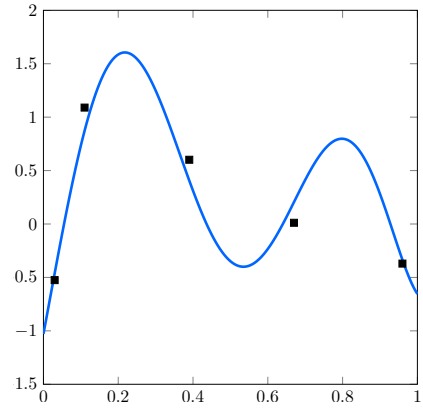
$$k(\boldsymbol{x}, \boldsymbol{x}') = \gamma^2 \exp\Big[-0.5 \sum_d (x_d - x'_d)^2 / l_d^2\Big]. \tag{3.26}$$

Therefore the set of model hyperparameters is $\{\lambda, \gamma, l_1, ..., l_D, \sigma\}$, a broad Gaussian hyperprior is placed on λ and uninformative Gamma priors are used for the other hyperparameters.

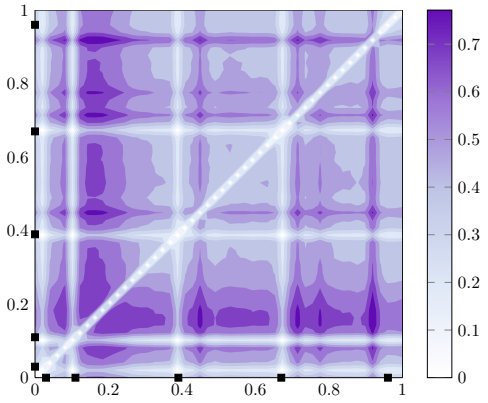
3.4.1 Assessing the Quality of the PPES Approximation

It is worth investigating how well $\hat{a}_{\text{PPES}}$ (3.23) is able to approximate a_{PPES} (3.5). In order to test the approximation in a manner amenable to visualisation, we generate a sample f from a Gaussian process prior on $\mathcal{X} = [0, 1]$, with $\gamma^2 = 1$, $\sigma^2 = 10^{-4}$ and $l^2 = 0.025$, and consider batches of size $Q = 2$. We set $M = 200$. A rejection sampling based approach is used to compute the ground truth a_{PPES}, defined on $\mathcal{X}^Q = [0, 1]^2$, implemented as follows. We first discretise $[0, 1]^2$, and sample $p(\boldsymbol{x}^*|\mathcal{D})$ in (3.5) by evaluating samples from $p(f|\mathcal{D})$ on the discrete points and choosing the input with highest function value. Further samples from $p(f|\mathcal{D})$ are evaluated on discrete points in $[0, 1]^2$ and rejected if the highest function value occurs not at $\boldsymbol{x}^*$. The non rejected samples are therefore samples from $p(f|\mathcal{D}, \boldsymbol{x}^*)$. We add independent Gaussian noise with variance σ^2 to these non rejected samples from the previous step and approximate $\mathrm{H}\Big[p\Big(y_1, y_2|\mathcal{D}, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}^*\Big)\Big]$ using kernel density estimation [Ahmad and Lin, 1976].

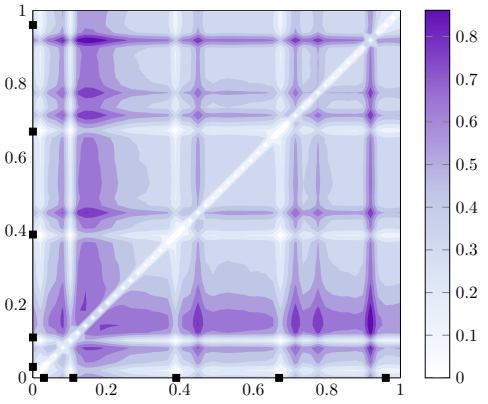
Figure 3.1 includes illustrations of (a) the objective function to be maximised, f, with 5 noisy observations, (b) the a_{PPES} ground truth obtained using the rejection sampling method and finally (c) $\hat{a}_{\text{PPES}}$ using the EP method we develop in the previous section. The black squares on the axes of Figures 3.1(b) and 3.1(c) represent the locations in $\mathcal{X} = [0, 1]$ where f has been noisily sampled, and the darker the shade, the larger the function value. The lightly shaded horizontal and vertical lines in these figures correspond to



(a) Synthetic function



(b) $a_{\mathrm{PPES}}(x, x')$



(c) $\hat{a}_{\mathrm{PPES}}(x, x')$

Fig. 3.1 Assessing the quality of our approximations to the parallel predictive entropy search strategy. (a) Synthetic objective function (blue line) defined on $[0, 1]$, with noisy observations (black squares). (b) Ground truth a_{PPES} defined on $[0, 1]^2$, obtained by rejection sampling. (c) Our approximation $\hat{a}_{\mathrm{PPES}}$ using expectation propagation. x and x' are two points in a batch. Dark regions correspond to pairs (x, x') with high utility, whilst faint regions correspond to pairs (x, x') with low utility, with respect to expected information gain.

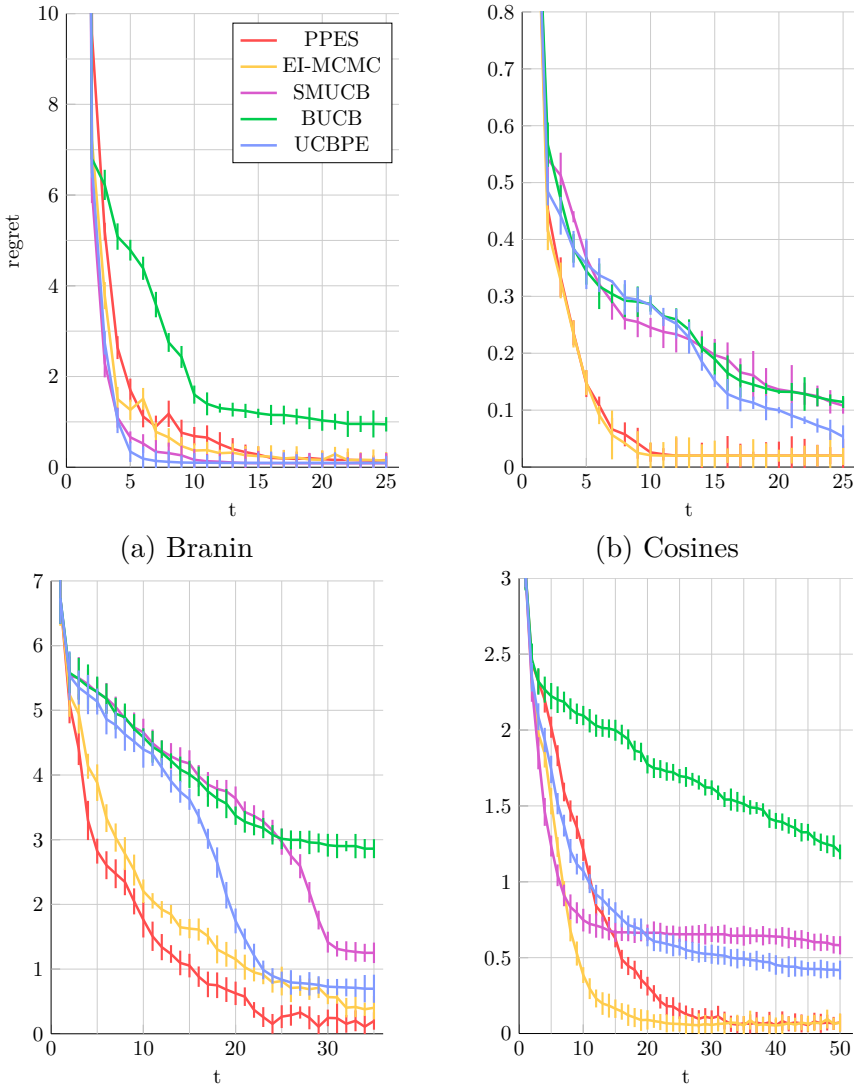
input values where the function has been evaluated already, where little information is to be gained from reevaluating, given that the data is not very noisy.

The figures representing a_{PPES} and $\hat{a}_{\mathrm{PPES}}$ appear to be symmetric, as is expected, since the set $\mathcal{S}_t = \{x, x'\}$ is not an ordered set, since all points in the set are probed in parallel i.e. $\mathcal{S}_t = \{x, x'\} = \{x', x\}$. The surface of $\hat{a}_{\mathrm{PPES}}$

is similar to that of a_{PPES}. In particular, the $\hat{a}_{\text{PPES}}$ approximation often appeared to be an *annealed* version of the ground truth a_{PPES}, in the sense that peaks were more pronounced, and non-peak areas were flatter. Since we are interested in $\operatorname{argmax}_{\{x,x'\}\in\mathcal{X}^2} a_{\text{PPES}}(\{x, x'\})$, our key concern is that the peaks of $\hat{a}_{\text{PPES}}$ occur at the same input locations as a_{PPES}. This appears to be the case in our experiment, suggesting that the $\operatorname{argmax} \hat{a}_{\text{PPES}}$ is a good approximation for $\operatorname{argmax} a_{\text{PPES}}$. Hernández-Lobato et al. [2014] found that a similar EP based approximation to the $Q = 1$ problem also matched the true predictive entropy well in regions which the user is most interested in.

We now test the performance of PPES in the task of finding the optimum of various objective functions. For each experiment, we compare PPES ($M = 200$) to EI-MC (with 100 MCMC samples), simulated matching with a UCB baseline policy, GP-BUCB and GP-UCB-PE. We use the random features method to sample from $p(\boldsymbol{x}^*|\mathcal{D})$, rejecting samples which led to failed EP runs. EP tended to fail when samples from $p(\boldsymbol{x}^*|\mathcal{D})$ were significantly below the predictive mean of the Gaussian process at query locations; this put a lot of pressure on the approximate EP Gaussian factors to learn low valued means, $\tilde{\mu}_q$, and small variance, $\tilde{\tau}_q$, to force the approximate Gaussian posterior to significantly lower its prediction of the function value conditioned on $\boldsymbol{x}^*$. The low values of $\tilde{\tau}_q$ led to poorly conditioned approximate covariance matrices, $\boldsymbol{\Sigma}$, which in turn led to failed matrix inverse and determinant operations. To improve stability, we forced $\tilde{\tau}_q$ to be at least 0.0001 on each run, however, we occasionally still experienced some failed EP runs.

An experiment of an objective function, f, consists of sampling 5 input points sampled uniformly at random and running each algorithm starting with these samples and their corresponding (noisy) function values. We measure performance after t batch evaluations using *immediate regret*, $r_t = |f(\tilde{\boldsymbol{x}}_t) - f(\boldsymbol{x}^*)|$, where $\boldsymbol{x}^*$ is the known optimiser of f and $\tilde{\boldsymbol{x}}_t$ is the recommendation of an algorithm after t batch evaluations. We perform 100 experiments for each objective function, and report the median of the immediate regret obtained for each algorithm. The confidence bands represent one standard deviation obtained from bootstrapping. The empirical



(a) Branin (b) Cosines

(c) Shekel (d) Hartmann

Fig. 3.2 Median of the immediate regret of the PPES, EI-MC, SMUCB, BUCB and UCBPE algorithms over 100 experiments on benchmark synthetic objective functions, using batches of size $Q = 3$.

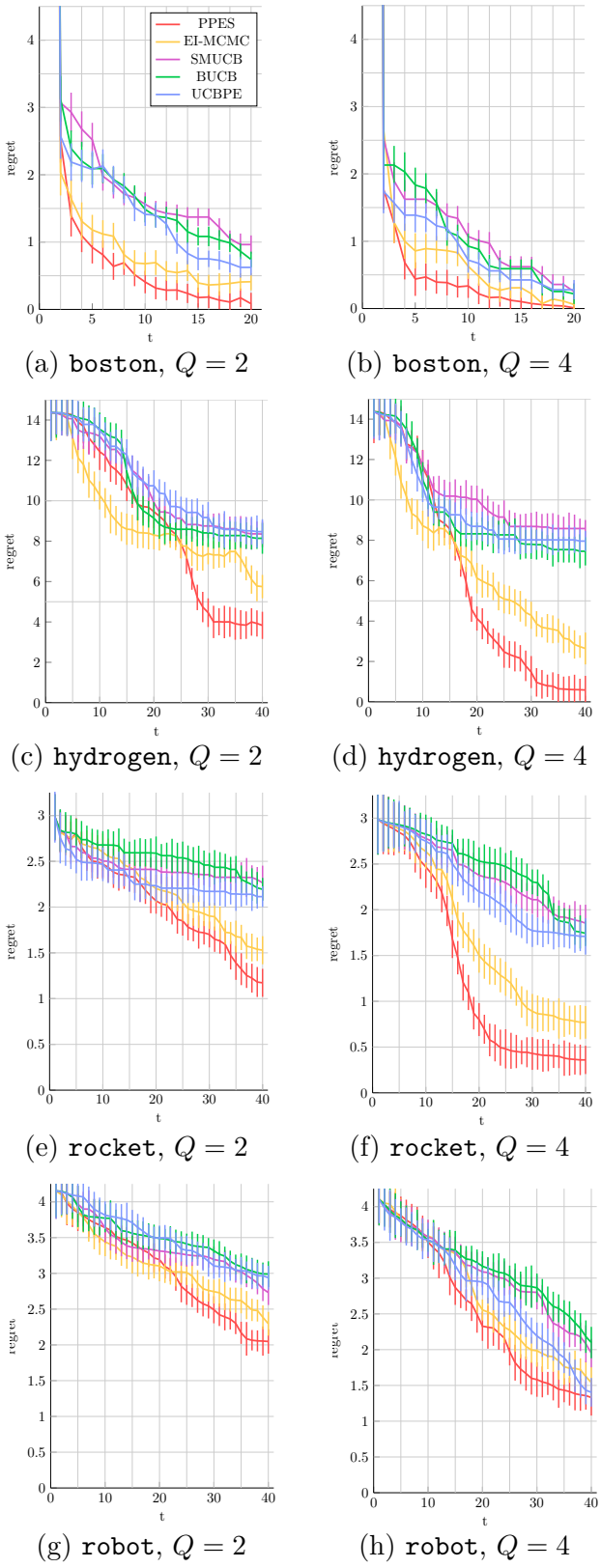
distribution of the immediate regret is heavy tailed, making the median more representative of where most data points lie than the mean.

3.4.2 Comparison of Methods on Synthetic Objectives

Our first set of experiments is on a set of synthetic benchmark objective functions including Branin-Hoo [Lizotte, 2008], a mixture of cosines [Anderson et al., 2000], a Shekel function with 10 modes [Shekel, 1971] (each defined on $[0, 1]^2$) and the Hartmann-6 function [Lizotte, 2008] (defined on $[0, 1]^6$). We choose batches of size $Q = 3$ at each decision time. The plots in Figure 3.2 illustrate the median immediate regrets found for each algorithm. The results suggest that the PPES algorithm performs close to best if not the best for each problem considered. EI-MC does significantly better on the Hartmann function, which is a relatively smooth function with very few modes, where greedy search appears beneficial. Entropy-based strategies are more exploratory in higher dimensions. Nevertheless, PPES does significantly better than GP-UCB-PE on 3 of the 4 problems, suggesting that our non-greedy batch selection procedure enhances performance versus a greedy entropy based policy.

3.4.3 Comparison of Methods on Real World Datasets or Simulations

We next consider maximisation of real world objective functions. The first, `boston`, returns the negative of the prediction error of a neural network trained on a random train/text split of the Boston Housing dataset [Bache and Lichman, 2013]. The weight-decay parameter and number of training iterations for the neural network are the parameters to be optimised over. The next function, `hydrogen`, returns the amount of hydrogen produced by particular bacteria as a function of pH and nitrogen levels of a growth medium [Burrows et al., 2009]. Thirdly we consider a function, `rocket`, which runs a simulation of a rocket [Hasbun, 2008] being launched from the Earth's surface and returns the time taken for the rocket to land on the Earth's surface. The variables to be optimised over are the launch height from the surface, the mass of fuel to use and the angle of launch with respect to the Earth's surface. If the rocket does not return, the function returns 0. Finally we consider a function, `robot`, which returns the walking speed of a bipedal robot [Westervelt and Grizzle, 2007]. The function's input



(a) `boston`, $Q = 2$ (b) `boston`, $Q = 4$

(c) `hydrogen`, $Q = 2$ (d) `hydrogen`, $Q = 4$

(e) `rocket`, $Q = 2$ (f) `rocket`, $Q = 4$

(g) `robot`, $Q = 2$ (h) `robot`, $Q = 4$

Fig. 3.3 Median of the immediate regret of the PPES and 4 other algorithms over 100 experiments on real world objective functions. Figures in the top row use batches of size $Q = 2$, whilst figures on the bottom row use batches of size $Q = 4$.

parameters, which live in $[0, 1]^8$, are the robot's controller. We add Gaussian noise with $\sigma = 0.1$ to the noiseless function. Note that all of the functions we consider are not available analytically. `boston` trains a neural network and returns test error, whilst `rocket` and `robot` run physical simulations involving differential equations before returning a desired quantity. Since the hydrogen dataset is available only for discrete points, we define `hydrogen` to return the predictive mean of a Gaussian process trained on the dataset.

Figure 3.3 show the median values of immediate regret by each method over 200 random initialisations. We consider batches of size $Q = 2$ and $Q = 4$. We find that PPES consistently outperforms competing methods on the functions considered. The greediness and nonrequirement of MCMC sampling of the SM-UCB, GP-BUCB and GP-UCB-PE algorithms make them amenable to large batch experiments, for example, [Desautels et al., 2012] consider optimisation in $\mathbb{R}^{45}$ with batches of size 10. However, these three algorithms all perform poorly when selecting batches of smaller size. The performance on the `hydrogen` function illustrates an interesting phenomena; whilst the immediate regret of PPES is mediocre initially, it drops rapidly as more batches are evaluated.

This behaviour is likely due to the non-greediness of the approach we have taken. EI-MC makes good initial progress, but then fails to explore the input space as well as PPES is able to. Recall that after each batch evaluation, an algorithm is required to output $\tilde{\boldsymbol{x}}_t$, its best estimate for the maximiser of the objective function. We observed that whilst competing algorithms tended to evaluate points which had high objective function values compared to PPES, yet when it came to recommending $\tilde{\boldsymbol{x}}_t$, PPES tended to do a better job. Our belief is that this occurred exactly because the PPES objective aims to maximise information gain rather than objective function value improvement.

The `rocket` function has a strong discontinuity making if difficult to maximise. If the fuel mass, launch height and/or angle are too high, the rocket would not return to the Earth's surface, resulting in a 0 function value. It can be argued that a stationary kernel Gaussian process is a poor model for this function, yet it is worth investigating the performance of a GP based models

since a practitioner may not know whether or not their black-box function is smooth apriori. PPES seemed to handle this function best and had fewer samples which resulted in 0 function value than each of the competing methods and made fewer recommendations which led to a 0 function value. The relative increase in PPES performance from increasing batch size from $Q = 2$ to $Q = 4$ is small for the `robot` function compared to the other functions considered. We believe this is a consequence of PPES's tendency to be more exploratory in higher dimensions than the alternative methods and the fact that we are using a slightly suboptimal optimisation procedure to save computation time. Our optimisation procedure first computes $\hat{a}_{\text{PPES}}$ at 1000 points selected uniformly at random, and performs gradient ascent from the best point. Since $\hat{a}_{\text{PPES}}$ is defined on $\mathcal{X}^Q = [0, 1]^{32}$, this method may miss a global optimum. Other methods all select their batches greedily, and hence only need to optimise in $\mathcal{X} = [0, 1]^8$. However, this should easily be avoided by using a more exhaustive gradient based optimiser, if a practitioner is willing to invest more time in the search loop.

3.5 Conclusions

We have developed parallel predictive entropy search, an information theoretic approach to batch Bayesian optimisation. Our method is greedy in the sense that it aims to maximise the one-step information gain about the location of $\boldsymbol{x}^*$, but it is not greedy in how it selects a set of points to evaluate next. Previous methods are doubly greedy, in that they look one step ahead, and also select a batch of points greedily. Competing methods are prone to under exploring, which hurts their performance on multi-modal, noisy objective functions, as we demonstrate in our experiments.

There are several issues which may be addressed in future work. Firstly, whilst in theory it is possible to find the global maximiser of $\hat{a}_{\text{PPES}}$, in practice it can be difficult and expensive to find. It may be interesting to consider entropy based batch optimisation procedures which do not require a global optimisation in the inner loop of the Bayesian optimisation procedure. If you look closely at the final expression we use to define $\hat{a}_{\text{PPES}}$, our approach

to deal with the hyperparameters $\boldsymbol{\psi}$ actually approximates the following function

$$\tilde{\alpha}(\mathcal{S}_t|\mathcal{D}) = \mathbb{E}_{p(\boldsymbol{\psi}|\mathcal{D})}\Big[\mathrm{H}\Big[\{y_q\}_{q=1}^{Q}|\mathcal{D},\mathcal{S}_t,\boldsymbol{\psi}\Big] - \mathbb{E}_{p(\boldsymbol{x}^*|\mathcal{D})}\Big[\mathrm{H}\Big[\{y_q\}_{q=1}^{Q}|\mathcal{D},\mathcal{S}_t,\boldsymbol{\psi},\boldsymbol{x}^*\Big]\Big]\Big].$$

What we would actually want is to marginalise out the Gaussian process and all hyperparameters before computing entropies, as in the following equation

$$\begin{aligned}\alpha(\mathcal{S}_t|\mathcal{D}) = \mathrm{H}\Big[\mathbb{E}_{p(\boldsymbol{\psi}|\mathcal{D})}\Big[\{y_q\}_{q=1}^{Q}|\mathcal{D},\mathcal{S}_t,\boldsymbol{\psi}\Big]\Big] \\ - \mathbb{E}_{p(\boldsymbol{x}^*|\mathcal{D})}\Big[\mathrm{H}\Big[\mathbb{E}_{p(\boldsymbol{\psi}|\mathcal{D})}\Big[\{y_q\}_{q=1}^{Q}|\mathcal{D},\mathcal{S}_t,\boldsymbol{\psi},\boldsymbol{x}^*\Big]\Big]\Big].\end{aligned}$$

The problem with the more desirable approach is that it would lead to having to compute the entropy of a mixture of Gaussians, which is no longer analytically tractable. Nevertheless, it would be interesting to study other ways of dealing with model hyperparameters.

Chapter 4

Predictive Entropy Search for Multi-Objective Optimisation

In this chapter, I describe an approach to extend the predictive entropy search framework to multi-objective optimisation. Most of this work was done in collaboration with Daniel Hernández-Lobato, José Miguel Hernández-Lobato and Ryan P. Adams and has been published at ICML [Hernández-Lobato et al., 2016]. Whilst the original idea and mathematical derivations of using expectation propagation and predictive entropy search for multi-objective optimisation analogous to PPES were my own, most of the implementation and experimentation was done by Daniel and Miguel.

4.1 Introduction

Most engineering problems require making design choices which aim to simultaneously optimise multiple objectives. For example, in designing a new drug, a pharmaceutical scientist may strive to simultaneously maximise the likelihood of curing an illness, minimise the chance of unwanted side-effects and minimise the cost of drug development. Typically there would not exist a particular option for which each objective is fully optimised. Subsequently, the scientist would wish to consider a range of options which trade off the multiple objectives. In a complex robotic system, we may be interested in minimising the energy consumption while maximising locomotion speed Ariizumi et al. [2014]. The ultimate choice should be *Pareto optimal*; there

should not exist an alternative option which can improve on the chosen option in every objective simultaneously.

The objective functions are often *unknown*, and can only be ascertained through pointwise evaluation. However, it can be *expensive* to evaluate these objectives, in the sense that they may necessitate large computational, economical or other resource. The challenge is to find a set of Pareto optimal points in as few sequential evaluations of the multiple objective functions as possible, so as to minimise the total expense.

A Bayesian theoretic approach to this task would be to probabilistically model the multiple unknown objective functions. Where the function should be evaluated next is decided by maximising the expected value of a chosen *acquisition*, or utility function, based on the posterior distribution of the objective functions given evaluations. Promising results have been shown using a Gaussian process to approximate each objective function [Knowles, 2006; Emmerich, 2008; Ponweiser et al., 2008; Picheny, 2015]. The model-based approach contrasts with model-free methods based on genetic algorithms or evolutionary strategies that are known to be effective for approximating the Pareto set, but demand a large number of function evaluations [Deb et al., 2002; Li, 2003; Zitzler and Thiele, 1999a].

Despite these successes, there are notable limitations to current model-based approaches: 1) they often build the acquisition function by transforming the multi-objective problem into a single-objective problem using scalarisation techniques (an approach that is expected to be suboptimal), 2) the acquisition function generally requires the evaluation of *all* of the objective functions at the same location in each iteration, and 3) the computational cost of evaluating the acquisition function typically grows exponentially with the number of objectives, which limits their applicability to optimisation problems with just 2 or 3 objectives.

In this chapter, we describe a strategy for multi-objective optimisation which addresses the concerns above. It extends the concept of predictive entropy search, by choosing a point to evaluate at next which is expected to

reduce the entropy of the Pareto set the most. The proposed approach is called *predictive entropy search for multi-objective optimisation* (PESMO). Several experiments involving real-world and synthetic optimisation problems, show that PESMO can lead to better performance than related methods from the literature. Furthermore, in PESMO the acquisition function is expressed as a sum across the different objectives, allowing for *decoupled* scenarios in which we can choose to only evaluate a subset of objectives at any given location. In the robotics example, one might be able to decouple the problems by estimating energy consumption from a simulator even if the locomotion speed could only be evaluated via physical experimentation. Another example, inspired by Gelbart et al. [2014], might be the design of a low-calorie cookie: one wishes to maximise taste while minimising calories, but calories are a simple function of the ingredients, while taste could require human trials. The results obtained show that PESMO can obtain better results with a smaller number of evaluations of the objective functions in such scenarios. Furthermore, we have observed that the decoupled evaluation provides significant improvements over a coupled evaluation when the number of objectives is large. Finally, unlike other methods [Ponweiser et al., 2008; Picheny, 2015], the computational cost of PESMO grows linearly with the number of objectives.

Problem Formulation

Our aim is to jointly maximise $L \geq 2$ bounded objectives $f_l : \mathcal{X} \to \mathbb{R}$ for $l = 1, ..., L$. Concretely, we wish to find a set of *Pareto efficient* points. Given distinct $\boldsymbol{y}_i \in \mathbb{R}^L$ for $i = 1, ..., n$, we write $\boldsymbol{y}_j \succeq \boldsymbol{y}_l$ when $y_{j,l} \geq y_{k,l}$ for each $l = 1, ..., L$, and say "$\boldsymbol{y}_j$ *dominates* $\boldsymbol{y}_l$". For the set of distinct points $\mathcal{Y} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_n\}$, the subset of Pareto efficient points, $\mathcal{P}(\mathcal{Y}) \subseteq \mathcal{Y}$, is defined as

$$\mathcal{P}(\mathcal{Y}) \equiv \Big\{\boldsymbol{y}_i \in \mathcal{Y} : \boldsymbol{y}_j \not\succeq \boldsymbol{y}_i \, \forall \boldsymbol{y}_j \in \mathcal{Y} \backslash \{\boldsymbol{y}_i\}\Big\}. \tag{4.1}$$

In other words, the Pareto efficient subset is the subset of all non-dominated points, and is always non empty. A dominated point, by definition, is a suboptimal choice since there exists a point which achieves a higher value for each of the L objectives. A measure of the quality of a set of Pareto efficient points is the volume of the set of points which the Pareto efficient

points dominate. This is defined more formally in the next chapter.

In a Bayesian optimisation setting, input locations $\boldsymbol{x}_1, \boldsymbol{x}_2, ... \in \mathcal{X}$, at which the expensive objective functions are evaluated, are chosen sequentially. Given function evaluations $\boldsymbol{y}_s = [f_1(\boldsymbol{x}_s), ..., f_L(\boldsymbol{x}_s)]^\top$ for $s = 1, ..., t$, $\boldsymbol{x}_{t+1} \in \mathcal{X}$ is chosen with the goal of improving the set of Pareto points as much as possible with as few future function evaluations. However, in its current frame, this is a qualitative goal and not a quantitative formulation. In the case of single objective Bayesian optimisation of a function $f : \mathcal{X} \to \mathbb{R}$, approaches for where to evaluate the function next are (a) where it is expected to most improve upon the current best value (expected improvement), (b) at a location which maximises a linear combination of the predictive mean and standard deviation (UCB) [Brochu et al., 2009], or (c) at a location which maximises information gain about the maximiser (predictive entropy search). In this work we focus on the entropy based ideas, following their success in the single-objective case. How can this quantitative single objective framework be generalised to the multi-objective case? This work, to the best of our knowledge, is the first attempt to apply entropy based ideas to multi-objective Bayesian optimisation.

4.2 Multi-objective Bayesian Optimisation via Predictive Entropy Search

In this section we describe the proposed approach for multi-objective optimisation based on *predictive entropy search*. Define the range set,

$$\mathcal{R} \equiv \{\boldsymbol{y} \in \mathbb{R}^L : \exists\, \boldsymbol{x} \in \mathcal{X} \text{ such that } y_l = f_l(\boldsymbol{x}) \text{ for } l = 1, ..., L\}, \tag{4.2}$$

for the functions $f_l(\cdot)$, with Pareto efficient subset $\mathcal{P}(\mathcal{R})$. We can therefore define the set of Pareto efficient inputs as

$$\mathcal{X}^* \equiv \{\boldsymbol{x} \in \mathcal{X} : \exists\, \boldsymbol{y}^* \in \mathcal{P}(\mathcal{R}) \text{ such that } y_l^* = f_l(\boldsymbol{x}) \text{ for } l = 1, ..., L\}, \tag{4.3}$$

which is the set we are interested in finding. Given some previous evaluations of each objective function $f_l(\cdot)$, we seek to choose new evaluations that

maximise the information gained about the Pareto set $\mathcal{X}^\star$. This approach requires a probabilistic model for the unknown objectives, and we therefore assume that each $f_l(\cdot)$ follows a Gaussian process (GP) prior [Rasmussen and Williams, 2006], with observation noise that is i.i.d. Gaussian with zero mean. For simplicity, we initially consider a coupled setting in which we evaluate all objectives at the same location in any given iteration. Nevertheless, the approach described can be easily extended to the *decoupled* scenario.

Let $\mathcal{D} \equiv \{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^N$ be the data (function evaluations) collected up to step N, where $\boldsymbol{y}_n$ is a L-dimensional vector with the values resulting from the evaluation of all objectives at step n, and $\boldsymbol{x}_n$ is a vector in input space denoting the evaluation location. The next query $\boldsymbol{x}_{N+1}$ is the one that maximises the expected reduction in the entropy $\mathrm{H}(\cdot)$ of the posterior distribution over the Pareto set $\mathcal{X}^\star$, i.e., $p(\mathcal{X}^\star|\mathcal{D})$. The acquisition function of PESMO is hence:

$$\alpha(\boldsymbol{x}|\mathcal{D}) = \mathrm{H}\Big[\mathcal{X}^\star|\mathcal{D}\Big] - \mathbb{E}_{p(\boldsymbol{y}|\mathcal{D},\boldsymbol{x})}\Big[\mathrm{H}\Big[\mathcal{X}^\star|\mathcal{D}\cup\{\boldsymbol{x},\boldsymbol{y}\}\Big]\Big], \tag{4.4}$$

where $\boldsymbol{y}$ is the output of all the GP models at $\boldsymbol{x}$ and the expectation is taken with respect to the posterior distribution for $\boldsymbol{y}$ given by, $p(\boldsymbol{y}|\mathcal{D},\boldsymbol{x}) = \prod_{l=1}^L p(y_l|\mathcal{D},\boldsymbol{x})$. The GPs are assumed to be independent *a priori*. This is a fairly strong assumption given that we expect the objectives we are jointly optimising to inherently be dependent on one another. In the next chapter, we explore the idea of explicitly modelling dependencies between objective functions, but in this work, to keep the math simple, we assume independence. The acquisition function in (4.4) is known as *entropy search* [Villemonteix et al., 2009; Hennig and Schuler, 2012]. Thus, at each iteration we set the location of the next evaluation to $\boldsymbol{x}_{N+1} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \alpha(\boldsymbol{x})$.

A practical difficulty, however, is that the exact evaluation of (4.4) is generally infeasible and the function must be approximated; we follow the approach described in Hernández-Lobato et al. [2014]; Houlsby et al. [2012]. In particular, (4.4) is the mutual information between $\mathcal{X}^\star$ and $\boldsymbol{y}$ given $\mathcal{D}$. The mutual information is symmetric and hence we can exchange the roles

of the variables $\mathcal{X}^\star$ and $\boldsymbol{y}$, leading to an equivalent expression:

$$\alpha(\boldsymbol{x}|\mathcal{D}) = \mathrm{H}\Big[\boldsymbol{y}|\mathcal{D},\boldsymbol{x}\Big] - \mathbb{E}_{p(\mathcal{X}^\star|\mathcal{D})}\Big[H\Big[\boldsymbol{y}|\mathcal{D},\boldsymbol{x},\mathcal{X}^\star\Big]\Big], \tag{4.5}$$

where the expectation is now with respect to the posterior distribution for the Pareto set $\mathcal{X}^\star$ given the observed data, and $\mathrm{H}\Big[\boldsymbol{y}|\mathcal{D},\boldsymbol{x},\mathcal{X}^\star\Big]$ measures the entropy of $p(\boldsymbol{y}|\mathcal{D},\boldsymbol{x},\mathcal{X}^\star)$, *i.e.*, the predictive distribution for the objectives at $\boldsymbol{x}$ given $\mathcal{D}$ and conditioned to $\mathcal{X}^\star$ being the Pareto set of the objective functions. This alternative formulation is known as *predictive entropy search* [Hernández-Lobato et al., 2014] and it significantly simplifies the evaluation of the acquisition function $\alpha(\cdot)$. In particular, we no longer have to evaluate or approximate the entropy of the Pareto set, $\mathcal{X}^\star$, which may be quite difficult. The new acquisition function obtained in (4.5) favours the evaluation in the regions of the input space for which $\mathcal{X}^\star$ is more informative about $\boldsymbol{y}$. These are precisely also the regions in which $\boldsymbol{y}$ is more informative about $\mathcal{X}^\star$.

The first term in the r.h.s. of (4.5) is straight-forward to evaluate; it is simply the entropy of the predictive distribution $p(\boldsymbol{y}|\mathcal{D},\boldsymbol{x})$, which is a factorisable L-dimensional Gaussian distribution. Thus, we have that

$$H(\boldsymbol{y}|\mathcal{D},\boldsymbol{x}) = \frac{L}{2}\log(2\pi e) + \sum_{l=1}^{L} 0.5\log(v_l^{\mathrm{PD}})\,, \tag{4.6}$$

where v_l^{PD} is the predictive variance of $f_l(\cdot)$ at $\boldsymbol{x}$. The difficulty comes from the evaluation of the second term in the r.h.s. of (4.5), which is intractable and must be approximated; we follow Hernández-Lobato et al. [2014] and approximate the expectation using a Monte Carlo estimate of the Pareto set, $\mathcal{X}^\star$ given $\mathcal{D}$. This involves sampling several times the objective functions from their posterior distribution $p(f_1,\ldots,f_K|\mathcal{D})$. This step is done as in Hernández-Lobato et al. [2014] using random kernel features and linear models that accurately approximate the samples from $p(f_1,\ldots,f_L|\mathcal{D})$. In practice, we generate 10 samples from the posterior of each objective $f_l(\cdot)$.

Given the samples of the objectives, we must optimise them to obtain a sample from the Pareto set $\mathcal{X}^\star$. Note that unlike the true objectives, the sampled functions can be evaluated without significant cost. Thus, given

these functions, we use a grid search with $d \times 1,000$ points to solve the corresponding multi-objective problem to find $\mathcal{X}^\star$, where d is the number of dimensions. Of course, in high dimensional problems such a grid search is expected to be sub-optimal; in that case, we use the NSGA-II evolutionary algorithm [Deb et al., 2002]. The Pareto set is then approximated using a representative subset of 50 points. Given such a sample of $\mathcal{X}^\star$, the differential entropy of $p(\boldsymbol{y}|\mathcal{D}, \boldsymbol{x}, \mathcal{X}^\star)$ is estimated using the expectation propagation algorithm Minka [2001], as described in the proceeding section.

4.3 Approximating the Conditional Predictive with Expectation Propagation

To approximate the entropy of the conditional predictive distribution $p(\boldsymbol{y}|\mathcal{D}, \boldsymbol{x}, \mathcal{X}^\star)$ we consider the distribution $p(\mathcal{X}^\star|f_1, \ldots, f_L)$. In particular, $\mathcal{X}^\star$ is the Pareto set of $f_1, \ldots, f_K$ if and only if $\forall \boldsymbol{x}^\star \in \mathcal{X}^\star, \forall \boldsymbol{x}' \in \mathcal{X}, \exists\, l \in \{1, \ldots, L\}$ such that $f_l(\boldsymbol{x}^\star) \geq f_l(\boldsymbol{x}')$, assuming maximisation. That is, each point within the Pareto set has to be better or equal to any other point in the domain of the functions in at least one of the objectives. Let $\boldsymbol{f}$ be the set $\{f_1, \ldots, f_L\}$. The conditions just described can be translated into the following un-normalised distribution for $\mathcal{X}^\star$:

$$\begin{aligned} p(\mathcal{X}^\star|\boldsymbol{f}) &\propto \prod_{\boldsymbol{x}^\star \in \mathcal{X}^\star} \prod_{\boldsymbol{x}' \in \mathcal{X}} \left[1 - \prod_{l=1}^{L} \mathbb{I}\Big(f_l(\boldsymbol{x}^\star) \leq f_l(\boldsymbol{x}')\Big)\right] \\ &= \prod_{\boldsymbol{x}^\star \in \mathcal{X}^\star} \prod_{\boldsymbol{x}' \in \mathcal{X}} \psi(\boldsymbol{x}', \boldsymbol{x}^\star)\,, \end{aligned} \tag{4.7}$$

where $\psi(\boldsymbol{x}', \boldsymbol{x}^\star) = 1 - \prod_{l=1}^{L} \mathbb{I}\Big(f_l(\boldsymbol{x}^\star) \leq f_l(\boldsymbol{x}')\Big)$ and $\mathbb{I}(\cdot)$ is an indicator function. Thus, the r.h.s. of (4.7) is non-zero only for a valid Pareto set. Next, we note that in the noiseless case $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{f}) = \prod_{i=1}^{K} \delta(y_l - f_l(\boldsymbol{x}))$, where $\delta(\cdot)$ is the Dirac delta function; in the noisy case we simply replace the delta functions with Gaussians. We can hence write the unnormalised version

of $p(\boldsymbol{y}|\mathcal{D}, \boldsymbol{x}, \mathcal{X}^\star)$ as:

$$\begin{aligned} p(\boldsymbol{y}|\mathcal{D}, \boldsymbol{x}, \mathcal{X}^\star) &\propto \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{f}) p(\mathcal{X}^\star|\boldsymbol{f}) p(\boldsymbol{f}|\mathcal{D}) d\boldsymbol{f} \\ &\propto \int \prod_{l=1}^{L} \delta(y_l - f_l(\boldsymbol{x})) \prod_{\boldsymbol{x}^\star \in \mathcal{X}^\star} \psi(\boldsymbol{x}, \boldsymbol{x}^\star) \\ &\quad \times \prod_{\boldsymbol{x}' \in \mathcal{X} \setminus \{\boldsymbol{x}\}} \psi(\boldsymbol{x}', \boldsymbol{x}^\star)\, p(\boldsymbol{f}|\mathcal{D})\, d\boldsymbol{f}\,, \end{aligned} \tag{4.8}$$

where we have separated out the factors ψ that do not depend on $\boldsymbol{x}$, the point in which the acquisition function $\alpha(\cdot)$ is going to be evaluated. The approximation to the r.h.s. of (4.8) is obtained in two stages. First, we approximate $\mathcal{X}$ with the set $\tilde{\mathcal{X}} = \{\boldsymbol{x}_n\}_{n=1}^N \cup \mathcal{X}^\star \cup \{\boldsymbol{x}\}$, *i.e.*, the union of the input locations where the objective functions have been already evaluated, the current Pareto set and the candidate location $\boldsymbol{x}$ on which $\alpha(\cdot)$ should be evaluated. Then, we replace each non-Gaussian factor ψ with a corresponding approximate Gaussian factor $\tilde{\psi}$ whose parameters are found using expectation propagation (EP) Minka [2001]. That is,

$$\begin{aligned} \psi(\boldsymbol{x}', \boldsymbol{x}^\star) &= 1 - \prod_{l=1}^{L} \mathbb{I}\Big(f_l(\boldsymbol{x}^\star) \leq f_l(\boldsymbol{x}')\Big) \\ &\approx \tilde{\psi}(\boldsymbol{x}', \boldsymbol{x}^\star) = \tilde{Z} \prod_{l=1}^{L} \tilde{\phi}_l(\boldsymbol{x}', \boldsymbol{x}^\star)\,, \end{aligned} \tag{4.9}$$

where each approximate factor $\tilde{\phi}_l$ is a scaled two-dimensional Gaussian distribution. In particular, we set

$$\tilde{\phi}_l(\boldsymbol{x}', \boldsymbol{x}^\star) = \mathcal{N}(\boldsymbol{v}_l; \tilde{\boldsymbol{\mu}}_l, \tilde{\boldsymbol{\Sigma}}_l),$$

where we have defined $\boldsymbol{v}_l = (f_l(\boldsymbol{x}'), f_l(\boldsymbol{x}^\star))^\top$, and $\tilde{Z}, \tilde{\boldsymbol{\mu}}_l$ and $\tilde{\boldsymbol{\Sigma}}_l$ are parameters to be adjusted by EP, which refines each $\tilde{\psi}$ until convergence to enforce that it looks similar to the corresponding exact factor ψ [Minka, 2001]. The approximate factors $\tilde{\psi}$ that do not depend on the candidate input $\boldsymbol{x}$ are reused multiple times to evaluate the acquisition function $\alpha(\cdot)$, and they only have to be computed once. The $|\mathcal{X}^\star|$ factors that depend on $\boldsymbol{x}$ must be obtained relatively quickly to guarantee that $\alpha(\cdot)$ is not very expensive to evaluate. Thus, in practice we only update those factors once using EP, i.e.,

they are not refined until convergence. This condradicts the result of Seeger [2008] which requires convergence of EP parameters for the gradients with respect to inputs to not depend on EP parameters, however, in practice this did not seem to be a problem. More details on the EP algorithm are given in the next few subsections.

4.3.1 Constructing The Approximate Conditional Predictive

Suppose we have a sample of the Pareto set given observations $\mathcal{X}^* = \{\boldsymbol{x}_1^*, ..., \boldsymbol{x}_M^*\}$ of size M, and a set of N locations where we have observed function values $\hat{\mathcal{X}} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\} \subset \mathcal{D}$. Let's assume we have trained the parameters for the approximate factors $\tilde{\psi}$. Since the f_l are independent a posteriori, we describe how to construct its conditional predictive distribution. Let $\boldsymbol{f}_{l,+} = [f_l(\boldsymbol{x}_1), ..., f_l(\boldsymbol{x}_N), f_l(\boldsymbol{x}_1^*), ..., f_l(\boldsymbol{x}_M^*)]^\top$, and define

$$w(\boldsymbol{f}_{l,+}) \equiv p(\boldsymbol{f}_{l,+}|\mathcal{D}) \prod_{\boldsymbol{x}\in\hat{\mathcal{X}}\cup\mathcal{X}^*} \prod_{\boldsymbol{x}^*\in\mathcal{X}^*\setminus\{\boldsymbol{x}\}} \tilde{\phi}_l(\boldsymbol{x}, \boldsymbol{x}^*), \tag{4.10}$$

then,

$$p(\boldsymbol{f}_{l,+}|\mathcal{X}^*, \mathcal{D}) \approx \frac{w(\boldsymbol{f}_{l,+})}{\int w(\boldsymbol{g}) d\boldsymbol{g}}. \tag{4.11}$$

Let us enforce an ordering over the factors $\tilde{\phi}_l$ such that for $1 \leq j \leq M$, $\tilde{\phi}_{l,i,j}$ is $\tilde{\phi}_l(\boldsymbol{x}_i, \boldsymbol{x}_j^*)$ for $1 \leq i \leq N$, and is $\tilde{\phi}_l(\boldsymbol{x}_{i-N}^*, \boldsymbol{x}_j^*)$ for $N+1 \leq i \leq N+M$ and $i - N \neq j$. Note that $p(\boldsymbol{f}_{l,+}|\mathcal{D})$ is simply a Gaussian process posterior and is therefore a $(N+M)$-dimensional multivariate Gaussian with mean, $\boldsymbol{m}_{l,+}$ and covariance, $\boldsymbol{K}_{l,+}$. Since the product of Gaussian densities is a scaled Gaussian density, $w(\boldsymbol{f}_{l,+})$ is a scaled multivariate Gaussian with mean, $\boldsymbol{\mu}_{l,+}$, covariance, $\boldsymbol{\Sigma}_{l,+}$ and scale Z_l. If we set $\boldsymbol{C}_{i,j}$ to be a matrix of size $2 \times (N+M)$, with the i^{th} column of the first row and the j^{th} column of the second row having entries 1, with all remaining entries 0, then

$$\boldsymbol{\Sigma}_{l,+} = \left(\boldsymbol{K}_{l,+}^{-1} + \sum_{j=1}^{M} \sum_{i=1, i\neq j+N}^{N+M} \boldsymbol{C}_{i,j}^\top \tilde{\boldsymbol{\Sigma}}_{l,i,j}^{-1} \boldsymbol{C}_{i,j} \right)^{-1} \tag{4.12}$$

$$\boldsymbol{\mu}_{l,+} = \boldsymbol{\Sigma}_{l,+} \left(\boldsymbol{K}_{l,+}^{-1} \boldsymbol{m}_{l,+} + \sum_{j=1}^{M} \sum_{i=1, i\neq j+N}^{N+M} \boldsymbol{C}_{i,j}^\top \tilde{\boldsymbol{\Sigma}}_{l,i,j}^{-1} \tilde{\boldsymbol{\mu}}_{l,i,j} \right)^{-1}. \tag{4.13}$$

Define $\boldsymbol{f}_l = [f_l(\boldsymbol{x}_1), ..., f_l(\boldsymbol{x}_N)]$. Since the multivariate Gaussian is consistent under marginalisation, the posterior distribution $p(\boldsymbol{f}_l|\mathcal{X}^*, \mathcal{D})$ is simply multivariate Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$, where $\boldsymbol{\mu}$ is made of the first N entries of $\boldsymbol{\mu}_+$, and $\boldsymbol{\Sigma}$ is made up of the first N rows and columns of $\boldsymbol{\Sigma}_+$.

Finally, to make a prediction of the value of f_l at a new input location $\boldsymbol{x}'$, we again use the fact that the conditional predictive distribution of a multivariate Gaussian distribution is also a Gaussian distribution, such that $p(f_l(\boldsymbol{x}')|\mathcal{X}^*, \mathcal{D}, \boldsymbol{x}')$ is $\mathcal{N}(\boldsymbol{x}'; m_l^{\text{CPD}}, v_l^{\text{CPD}})$.

4.3.2 Computing the EP parameters

We now discuss how to update the parameters of $\tilde{\psi}(\boldsymbol{x}_i, \boldsymbol{x}_j^*)$. First compute the cavity distribution by removing all of the relevant factors,

$$w^{\backslash i,j}(\boldsymbol{f}_+) = \frac{w(\boldsymbol{f}_+)}{\tilde{Z}_{i,j} \prod_{l=1}^{L} \mathcal{N}\Big(\boldsymbol{f}_+; \boldsymbol{C}_{i,j}^\top \tilde{\boldsymbol{\mu}}_{l,i,j}, \boldsymbol{C}_{i,j}^\top \tilde{\boldsymbol{\Sigma}}_{l,i,j} \boldsymbol{C}_{i,j}\Big)} \tag{4.14}$$

and compute their Gaussian parameters. Since we are dividing a Gaussian p.d.f. by another Gaussian p.d.f. we have parameter updates given by

$$\boldsymbol{\Sigma}_{\backslash l,i,j} = \left(\boldsymbol{\Sigma}_{l,+}^{-1} - \boldsymbol{C}_{i,j}^\top \tilde{\boldsymbol{\Sigma}}_{l,i,j}^{-1} \boldsymbol{C}_{i,j}\right)^{-1} \tag{4.15}$$

$$\boldsymbol{\mu}_{\backslash l,i,j} = \boldsymbol{\Sigma}_{\backslash l,i,j}\left(\boldsymbol{\Sigma}_{l,+}^{-1}\boldsymbol{\mu}_{l,+} - \boldsymbol{C}_{i,j}^\top \tilde{\boldsymbol{\Sigma}}_{l,i,j}^{-1} \tilde{\boldsymbol{\mu}}_{l,i,j}\right). \tag{4.16}$$

The next step in EP inference is the projection step, where we match moments between $\psi(\boldsymbol{x}_i, \boldsymbol{x}_j^*) w^{\backslash i,j}(\boldsymbol{f}_+)$ and $\tilde{\psi}(\boldsymbol{x}_i, \boldsymbol{x}_j^*) w^{\backslash i,j}(\boldsymbol{f}_+)$. We use derivatives of the

logarithm of the zeroth moment [Minka, 2008] to compute the parameters

$$\begin{aligned}\hat{Z}_{i,j} &= \int \psi(\boldsymbol{x}_i, \boldsymbol{x}_j^*) w^{\setminus i,j}(\boldsymbol{f}_+) d\boldsymbol{f}_+ \\ &= 1 - \prod_{l=1}^{L} \Phi(\beta_{l,i,j}), \end{aligned} \tag{4.17}$$

$$\begin{aligned}\hat{\boldsymbol{\mu}}_{l,i,j} &= \boldsymbol{\mu}_{\setminus l,i,j} + \boldsymbol{\Sigma}_{\setminus l,i,j} \frac{\partial \log \hat{Z}_{i,j}}{\partial \boldsymbol{\mu}_{\setminus l,i,j}} \\ &= \boldsymbol{\mu}_{\setminus l,i,j} - \frac{\phi(\beta_{l,i,j})}{\alpha_{l,i,j} \hat{Z}_{i,j}} \boldsymbol{\Sigma}_{\setminus l,i,j} \boldsymbol{c}_{i,j}, \end{aligned} \tag{4.18}$$

$$\begin{aligned}\hat{\boldsymbol{\Sigma}}_{l,i,j} &= \boldsymbol{\Sigma}_{\setminus l,i,j} - \boldsymbol{\Sigma}_{\setminus l,i,j} \left(\left(\frac{\partial \log \hat{Z}_{i,j}}{\partial \boldsymbol{\mu}_{\setminus l,i,j}} \right) \left(\frac{\partial \log \hat{Z}_{i,j}}{\partial \boldsymbol{\mu}_{\setminus l,i,j}} \right)^\top - 2 \frac{\partial \log \hat{Z}_{i,j}}{\partial \boldsymbol{\Sigma}_{\setminus l,i,j}} \right) \boldsymbol{\Sigma}_{\setminus l,i,j} \\ &= \boldsymbol{\Sigma}_{\setminus l,i,j} - \left(\frac{\phi(\beta_{l,i,j})}{\alpha_{l,i,j} \hat{Z}_{i,j}} \right) \left(\frac{\phi(\beta_{l,i,j})}{\hat{Z}_{i,j}} - \beta_{l,i,j} \right) \left(\boldsymbol{\Sigma}_{\setminus l,i,j} \boldsymbol{c}_{i,j} \right) \left(\boldsymbol{\Sigma}_{\setminus l,i,j} \boldsymbol{c}_{i,j} \right)^\top, \end{aligned} \tag{4.19}$$

where $\alpha_{l,i,j} = \left(\boldsymbol{c}_{i,j}^\top \boldsymbol{\Sigma}_{\setminus l,i,j} \boldsymbol{c}_{i,j} \right)$, $\beta_{l,i,j} = \frac{\boldsymbol{c}_{i,j}^\top \boldsymbol{\mu}_{\setminus l,i,j}}{\sqrt{\alpha_{l,i,j}}}$, and $\boldsymbol{c}_{i,j}$ is a vector of length $N+M$ with i^{th} entry -1, j^{th} entry 1 and remaining entries 0. The projection step is completed by updating the site parameters as follows

$$\tilde{\boldsymbol{\Sigma}}_{l,i,j} = \left(\boldsymbol{C}_{i,j}^\top \left(\hat{\boldsymbol{\Sigma}}_{l,i,j}^{-1} - \boldsymbol{\Sigma}_{\setminus l,i,j}^{-1} \right) \boldsymbol{C}_{i,j} \right)^{-1} \tag{4.20}$$

$$\tilde{\boldsymbol{\mu}}_{l,i,j} = \tilde{\boldsymbol{\Sigma}}_{l,i,j} \boldsymbol{C}_{i,j}^\top \left(\hat{\boldsymbol{\Sigma}}_{l,i,j}^{-1} \hat{\boldsymbol{\mu}}_{l,i,j} - \boldsymbol{\Sigma}_{\setminus l,i,j}^{-1} \boldsymbol{\mu}_{\setminus l,i,j} \right) \tag{4.21}$$

$$\tilde{Z}_{i,j} = \hat{Z}_{i,j} \det\left(2\pi \boldsymbol{\Omega}_{i,j} \right)^{\frac{1}{2}} \exp\left(\frac{1}{2} \left(\tilde{\boldsymbol{\mu}}_{l,i,j} - \boldsymbol{\mu}_{\setminus l,i,j} \right)^\top \boldsymbol{\Omega}_{i,j}^{-1} \left(\tilde{\boldsymbol{\mu}}_{l,i,j} - \boldsymbol{\mu}_{\setminus l,i,j} \right) \right), \tag{4.22}$$

where $\boldsymbol{\Omega}_{i,j} \equiv \sum_{l=1}^{L} \left(\boldsymbol{\Sigma}_{\setminus l,i,j} + \boldsymbol{C}_{i,j}^\top \tilde{\boldsymbol{\Sigma}}_{l,i,j} \boldsymbol{C}_{i,j} \right)$. The computation of cavity distributions and projection is repeated until convergence.

4.3.3 Parallel EP Updates and Damping

The EP algorithm was implemented in a way such that the parameters of the approximate factors $\tilde{\psi}$ are updated in parallel. That is, the cavity distribution for each factor ψ and corresponding updates of $\tilde{\psi}$ are all computed in parallel.

The EP parameter updates also were damped, that is, the parameters of each updated factor are set to a linear combination of the old parameters and the new parameters. This approach prevents large changes to the parameter values and therefore helps convergence properties of the algorithm. Damping does not change the fixed points of EP.

4.3.4 Sampling the Posterior Pareto Set

Until now we have assumed the ability to sample from $p(\mathcal{X}^*|\mathcal{D})$, in this subsection we discuss how we do this in practice. We follow the method in Hernández-Lobato et al. [2014] and Shah and Ghahramani [2015] to approximately sample from $p(f_1, ..., f_l|\mathcal{D})$ using random kernel features and linear models. In practice, we generate 10 samples from the posterior of each objective $f_l(\cdot)$.

Given the samples, we optimise them to obtain a sample from $p(\mathcal{X}^*|\mathcal{D})$. The sampled functions can be evaluated without significant cost. Given the sampled functions, we use a grid search with $d \times 1000$ points to solve the corresponding multi-objective problem to find $\mathcal{X}^*$, where d is the number of dimensions.

4.3.5 The Resultant Approximation

Suppose we make the Gaussian approximations as in (4.9), we can approximate (4.5) with

$$\hat{\alpha}(\boldsymbol{x}) = \frac{1}{2}\sum_{l=1}^{L}\left(\log v_l^{\mathrm{PD}}(\boldsymbol{x}) - \frac{1}{S}\sum_{s=1}^{S}\log v_l^{\mathrm{CPD}}(\boldsymbol{x}|\mathcal{X}_{(s)}^{\star})\right), \tag{4.23}$$

where S is the number of Monte Carlo samples, $\{\mathcal{X}_{(s)}^{\star}\}_{s=1}^{S}$ are the Pareto sets sampled to approximate the expectation in (4.5), and $v_l^{\mathrm{PD}}(\boldsymbol{x})$ and $v_l^{\mathrm{CPD}}(\boldsymbol{x}|\mathcal{X}_{(s)}^{\star})$ are respectively the variances of the predictive distribution at $\boldsymbol{x}$, before and after conditioning on $\mathcal{X}_{(s)}^{\star}$. v_l^{CPD} is computed using the EP inference scheme decsribed in the previous subsections. Last, in the case of noisy observations around each $f_l(\cdot)$, we just increase the predictive variances by adding the noise variance. The next location to be tested is computed as

$\boldsymbol{x}_{N+1} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \hat{\alpha}(\boldsymbol{x})$.

Note that (4.23) is the sum of L functions

$$\hat{\alpha}_l(\boldsymbol{x}) = \frac{1}{2}\left(\log v_l^{\mathrm{PD}}(\boldsymbol{x}) - \frac{1}{S}\sum_{s=1}^{S} \log v_l^{\mathrm{CPD}}(\boldsymbol{x}|\mathcal{X}^\star_{(s)}) \right), \tag{4.24}$$

that intuitively measure the contribution of each objective to the total acquisition. In a *decoupled* evaluation setting, each $\hat{\alpha}_l(\cdot)$ can be individually maximised to identify the location $\boldsymbol{x}_l^{\mathrm{op}} = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \hat{\alpha}_l(\boldsymbol{x})$, on which it is expected to be most useful to evaluate each of the L objectives. The objective l with the largest individual acquisition $\hat{\alpha}_l(\boldsymbol{x}_l^{\mathrm{op}})$ can then be chosen for evaluation in the next iteration. This approach is expected to reduce the entropy of the posterior over the Pareto set more quickly, *i.e.*, with a smaller number of evaluations of the objectives, and to lead to better results.

The total computational cost of evaluating the acquisition function $\hat{\alpha}(\boldsymbol{x})$ includes the cost of running EP, which is $\mathcal{O}(Lm^3)$, where $m = N + |\mathcal{X}^\star_{(s)}|$, N is the number of observations made and L is the number of objectives. This is done once per each sample $\mathcal{X}^\star_{(s)}$. After this, we can re-use the factors that are independent of the candidate location $\boldsymbol{x}$. The cost of computing the predictive variance at each $\boldsymbol{x}$ is hence $\mathcal{O}(L|\mathcal{X}^\star_{(s)}|^3)$. In our experiments, the size of the Pareto set sample $\mathcal{X}^\star_{(s)}$ is 50, which means that m is a few hundred at most.

4.4 Related Work

ParEGO is another method for multi-objective Bayesian optimisation [Knowles, 2006]. ParEGO transforms the multi-objective problem into a single-objective problem using a scalarisation technique: at each iteration, a vector of L weights $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_L)^{\mathrm{T}}$, with $\theta_l \in [0, 1]$ and $\sum_{l=1}^{L} \theta_l = 1$, is sampled at random from a uniform distribution. Given $\boldsymbol{\theta}$, a single-objective function is

built:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \max_{l=1}^{L}(\theta_l f_l(\boldsymbol{x})) + \rho \sum_{l=1}^{L} \theta_l f_l(\boldsymbol{x}), \tag{4.25}$$

where ρ is set equal to 0.05. The non-linear part of the function guarantees that points in the non-convex regions of the Pareto front (this is simply the function space values associated to the Pareto set) can minimise the function, while the linear part of the function ensures that Pareto points are preferred over weak-Pareto points (points that are not strictly better in at least one objective, but better or equal). See Nakayama et al. [2009, Sec. 1.3.3] for further details. After step N of the optimisation process, and given $\boldsymbol{\theta}$, a new set of N observations of $f_{\boldsymbol{\theta}}(\cdot)$ are obtained by evaluating this function in the already observed points $\{\boldsymbol{x}_n\}_{n=1}^{N}$. Then, a GP model is fit to the new data and expected improvement [Mockus et al., 1978; Jones et al., 1998] is used find the location of the next evaluation $\boldsymbol{x}_{N+1}$. The cost of evaluating the acquisition function in ParEGO is $\mathcal{O}(N^3)$, where N is the number of observations made. This is the cost of fitting the GP to the new data (only done once). Thus, ParEGO is a simple technique that leads to a fast acquisition function. Nevertheless, it is often outperformed by more advanced approaches [Ponweiser et al., 2008].

SMSego is another technique for multi-objective Bayesian optimisation [Ponweiser et al., 2008]. The first step in SMSego is to find a set of Pareto points $\tilde{\mathcal{X}}^{\star}$, *e.g.*, by optimising the posterior means of the GPs, or by finding the non-dominated observations. Consider now an optimistic estimate of the objectives at input location $\boldsymbol{x}$ given by $m_l^{\mathrm{PD}}(\boldsymbol{x}) - c\sqrt{v_l^{\mathrm{PD}}(\boldsymbol{x})}$, where c is some constant, and $m_l^{\mathrm{PD}}(\boldsymbol{x})$ and $v_l^{\mathrm{PD}}(\boldsymbol{x})$ are the posterior mean and variance of the l^{th} objective at location $\boldsymbol{x}$, respectively. The acquisition value computed at a candidate location $\boldsymbol{x} \in \mathcal{X}$ by SMSego is given by the gain in hyper-volume obtained by the corresponding optimistic estimate, after an ϵ-correction has been made. The hyper-volume is simply the volume of points in functional space above the Pareto front (this is the function space values associated to the Pareto set), with respect to a given reference point [Zitzler and Thiele, 1999a]. Because the hyper-volume is maximised by the actual Pareto set, it is a natural measure of performance. Thus, SMSego does not reduce the

problem to a single-objective. However, at each iteration it has to find a set of Pareto points and to fit a different GP to each one of the objectives. This gives a computational cost that is $\mathcal{O}(LN^3)$. Finally, evaluating the gain in hyper-volume at each candidate location $\boldsymbol{x}$ is also more expensive than the computation of expected improvement in ParEGO. We discuss the Pareto hyper-volume in more detail in the next chapter.

A similar method to SMSego is the Pareto active learning (PAL) algorithm [Zuluaga et al., 2013]. At iteration N, PAL uses the GP prediction for each point $\boldsymbol{x} \in \mathcal{X}$ to maintain an uncertainty region $\mathcal{R}_N(\boldsymbol{x})$ about the objective values associated with $\boldsymbol{x}$. This region is defined as the intersection of $\mathcal{R}_{N-1}(\boldsymbol{x})$, *i.e.*, the uncertainty region in the previous iteration, and $\mathcal{Q}_c(\boldsymbol{x})$, defined as as the hyper-rectangle with lower-corner given by $m_l^{\mathrm{PD}}(\boldsymbol{x}) - c\sqrt{v_l^{\mathrm{PD}}(\boldsymbol{x})}$, for $l = 1, \ldots, L$, and upper-corner given by $m_l^{\mathrm{PD}}(\boldsymbol{x}) + c\sqrt{v_l^{\mathrm{PD}}(\boldsymbol{x})}$, for $l = 1, \ldots, L$, for some constant c. Given these regions, PAL classifies each point $\boldsymbol{x} \in \mathcal{X}$ as Pareto-optimal, non-Pareto-optimal or uncertain. A point is classified as Pareto-optimal if the worst value in $\mathcal{R}_N(\boldsymbol{x})$ is not dominated by the best value in $\mathcal{R}_N(\boldsymbol{x}')$, for any other $\boldsymbol{x}' \in \mathcal{X}$, with an ϵ tolerance. A point is classified as non-Pareto-optimal if the best value in $\mathcal{R}_N(\boldsymbol{x})$ is dominated by the worst value in $\mathcal{R}_N(\boldsymbol{x}')$ for any other $\boldsymbol{x}' \in \mathcal{X}$, with an ϵ tolerance. All other points remain uncertain. After the classification, PAL chooses the uncertain point $\boldsymbol{x}$ with the largest uncertainty region $\mathcal{R}_N(\boldsymbol{x})$. The total computational cost of PAL is hence similar to that of SMSego.

The expected hyper-volume improvement (EHI) [Emmerich, 2008] is a natural extension of expected improvement to the multi-objective setting Mockus et al. [1978]; Jones et al. [1998]. Given the predictive distribution of the GPs at a candidate input location $\boldsymbol{x}$, the acquisition is the expected increment of the hyper-volume of a candidate Pareto set $\tilde{\mathcal{X}}^\star$. Thus, EHI also needs to find a Pareto set $\tilde{\mathcal{X}}^\star$. This set can be obtained as in SMSego. A difficulty is, however, that computing the expected increment of the hyper-volume is very expensive. For this, the output space is divided in a series of cells, and the probability of improvement is simply obtained as the probability that the observation made at $\boldsymbol{x}$ lies in a non-dominated cell. This involves a sum across

all non-dominated cells, whose number grows exponentially with the number of objectives L. In particular, the total number of cells is $(|\tilde{\mathcal{X}}^\star| + 1)^L$. Thus, although some methods have been suggested to speed-up its calculation, *e.g.*, Hupkens et al. [2014], EHI is only feasible for a 2 or 3 objectives at most.

Sequential uncertainty reduction (SUR) is another method proposed for multi-objective Bayesian optimisation [Picheny, 2015]. The working principle of SUR is similar to that of EHI. However, SUR considers the probability of improving the hyper-volume in the whole domain of the objectives $\mathcal{X}$. Thus, SUR also needs to find a set of Pareto points $\tilde{\mathcal{X}}^\star$. These can be obtained as in SMSego. The acquisition computed by SUR is simply the expected decrease in the area under the probability of improving the hyper-volume, after evaluating the objectives at a new candidate location $\boldsymbol{x}$. The SUR acquisition is computed also by dividing the output space in a total of $(|\tilde{\mathcal{X}}^\star| + 1)^L$ cells, and the area under the probability of improvement is obtained using a Sobol sequence as the integration points. Although some grouping of the cells has been suggested [Picheny, 2015], SUR is an extremely expensive criterion that is only feasible for 2 or 3 objectives at most.

The proposed approach, PESMO, differs from the methods described in this section in that 1) it does not transform the multi-objective problem into a single-objective, 2) the acquisition function of PESMO can be decomposed as the sum of L individual acquisition functions, and this allows for decoupled evaluations, and 3) the computational cost of PESMO is linear in the total number of objectives L.

4.5 Experiments

We compare PESMO with the other strategies for multi-objective optimisation described in Section 4.4: ParEGO, SMSego, EHI and SUR. We do not compare results with PAL because it is expected to give similar results to those of SMSego, as both methods are based on a lower confidence bound. We have coded all these methods in the software for Bayesian optimisation

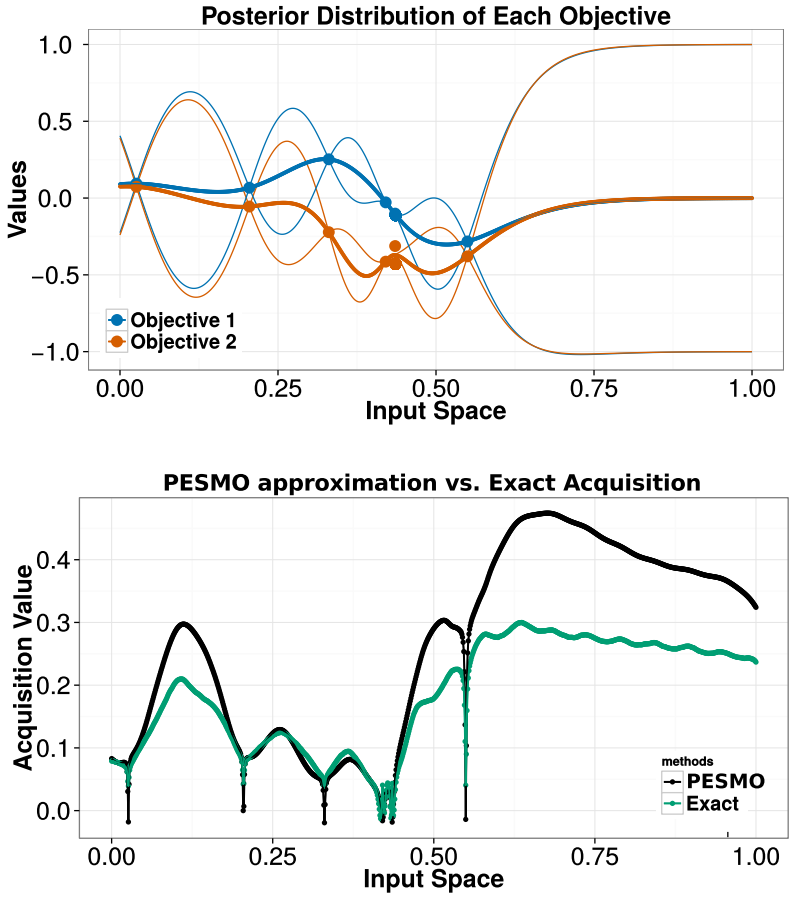


Fig. 4.1 **Top.** Observations of each objective and posterior mean and standard deviations of each GP model. Note that objectives are evaluated at the same input locations. **Bottom.** Estimates of the acquisition function ((4.5)) by PESMO, and by a Monte Carlo method combined with a non-parametric estimator of the entropy (Exact), which is expected to be more accurate.

Spearmint [1]. In all GP models we use a Matérn covariance function, and all hyper-parameters (noise, length-scales and amplitude) are approximately sampled from their posterior distribution (we generate 10 samples from this distribution). The acquisition function of each method is averaged over these samples, analogous to PPES from the last chapter. In ParEGO we consider a different scalarisation (*i.e.*, a different value of $\boldsymbol{\theta}$) for each sample of the hyper-parameters. In SMSego, EHI and SUR, for each hyper-parameter

[1] https://github.com/JasperSnoek/spearmint

sample we consider a different Pareto set $\tilde{\mathcal{X}}^\star$, obtained by optimising the posterior means of the GPs. The resulting Pareto set is extended by including all non-dominated observations. Finally, at iteration N, each method gives a recommendation in the form of a Pareto set obtained by optimising the posterior means of the GPs (we average the posterior means over the hyper-parameter samples). The acquisition function of each method is maximised using L-BFGS. A grid of size $1,000$ is used to find a good starting point for the optimisation process. The gradients of the acquisition function are approximated by differences (except in ParEGO).

4.5.1 Accuracy of the PESMO Approximation

One question to be experimentally addressed is whether the proposed approximations are sufficiently accurate for effective identification of the Pareto set. We compare in a one-dimensional problem with two objectives the acquisition function computed by PESMO with a more accurate estimate obtained via expensive Monte Carlo sampling and a non-parametric estimator of the entropy [Singh et al., 2003]. Figure 4.1 (top) shows at a given step the observed data and the posterior mean and the standard deviation of each of the two objectives. The figure on the bottom shows the corresponding acquisition function computed by PESMO and by the Monte Carlo method (Exact). We observe that both functions look very similar, including the location of the global maximiser. This indicates that (4.23), obtained by expectation propagation, is potentially a good approximation of (4.5), the exact acquisition function. The supplementary material has extra results that show that each of the individual acquisition functions computed by PESMO, *i.e.*, $\alpha_l(\cdot)$, for $l = 1, 2$, are also accurate.

Next we wish to see whether the proposed approximations for the individual acquisition functions $\alpha_l(\cdot)$, for $l = 1, \ldots, L$, with L the total number of objectives are sufficiently accurate in the decoupled case of PESMO. For this, we compare in a one-dimensional problem with two objectives the acquisition functions $\alpha_1(\cdot)$ and $\alpha_2(\cdot)$ computed by PESMO, with a more accurate estimate obtained via expensive Monte Carlo sampling and a non-parametric estimator of the entropy Singh et al. [2003]. This estimate measures the

expected decrease in the entropy of the predictive distribution of one of the objectives, at a given location of the input space, after conditioning to the Pareto set. Importantly, in the decoupled case, the observations corresponding to each objective need not be located at the same input locations.

Figure 4.2 (top) shows at a given step of the optimisation process, the observed data and the posterior mean and the standard deviation of each of the two objectives. The figure on the middle shows the corresponding acquisition function corresponding to the first objective, $\alpha_1(\cdot)$, computed by PESMO and by the Monte Carlo method (Exact). The figure on the bottom shows the same results for the acquisition function corresponding to the second objective, $\alpha_2(\cdot)$. We observe that both functions look very similar, including the location of the global maximiser. This indicates that the approximation obtained by expectation propagation is potentially good also in the decoupled setting.

4.5.2 Experiments with Synthetic Objectives

To initially compare PESMO with other approaches, we consider a 3-dimensional problem with 2 objectives obtained by sampling the functions from the corresponding GP prior. We generate 100 of these problems and report the average performance of each method, when considering noiseless observations and when the observations are contaminated with Gaussian noise with standard deviation equal to 0.1. The performance metric employed is the hyper-volume indicator, which is maximised by the actual Pareto set [Zitzler and Thiele, 1999a]. More precisely, at each iteration we report the logarithm of the relative difference between the hyper-volume of the actual Pareto set, which is obtained by optimising the actual objectives, and the hyper-volume of the recommendation, which is obtained by optimising the posterior means of the GPs. Figure 4.3 shows, as a function of the evaluations made, the average performance of each method with the corresponding error bars. PESMO obtains the best results, and when executed in a decoupled scenario slight improvements are observed, although only in the case of noisy observations.

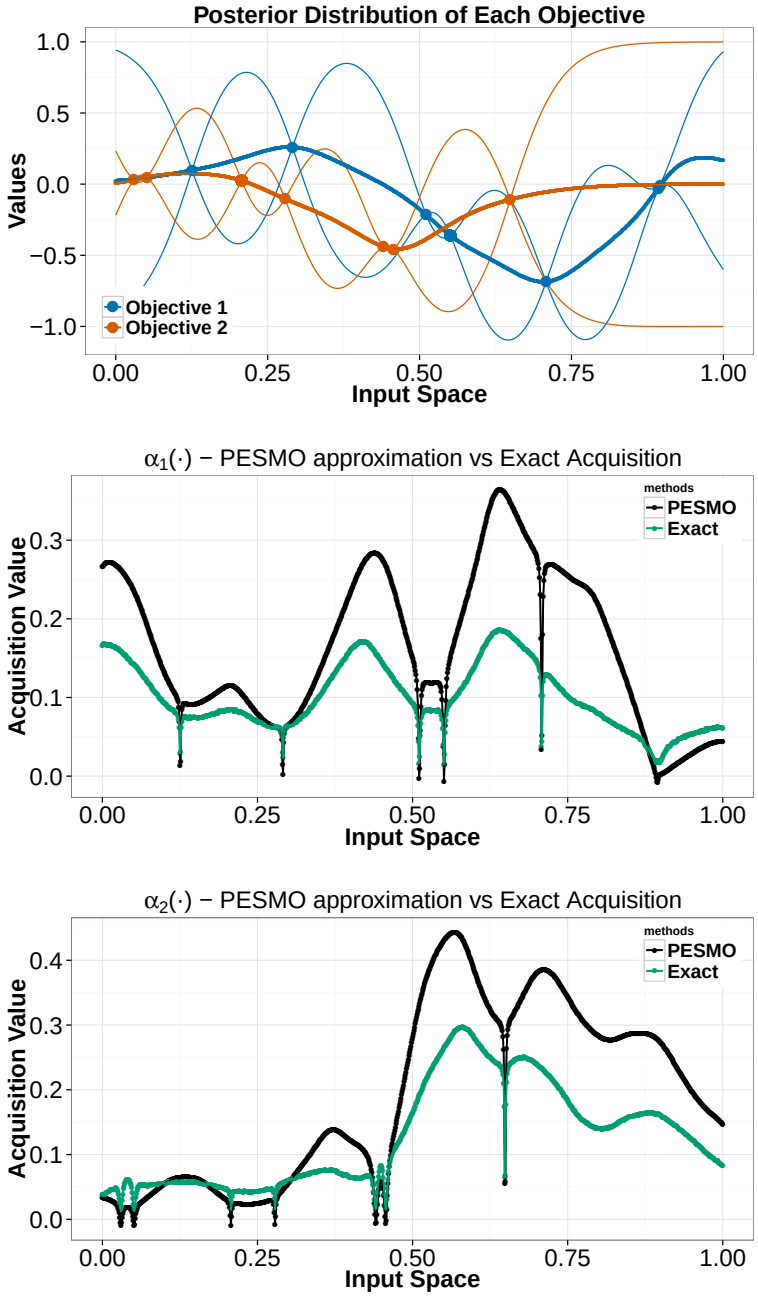


Fig. 4.2 **Top.** Observations of each objective and posterior mean and standard deviations of each GP model. **Middle.** Estimates of the acquisition function corresponding to the first objective, $\alpha_1(\dot{)}$, by PESMO, and by a Monte Carlo method combined with a non-parametric estimator of the entropy. **Bottom.** Same results for the acquisition function corresponding to the second objective $\alpha_2(\cdot)$.

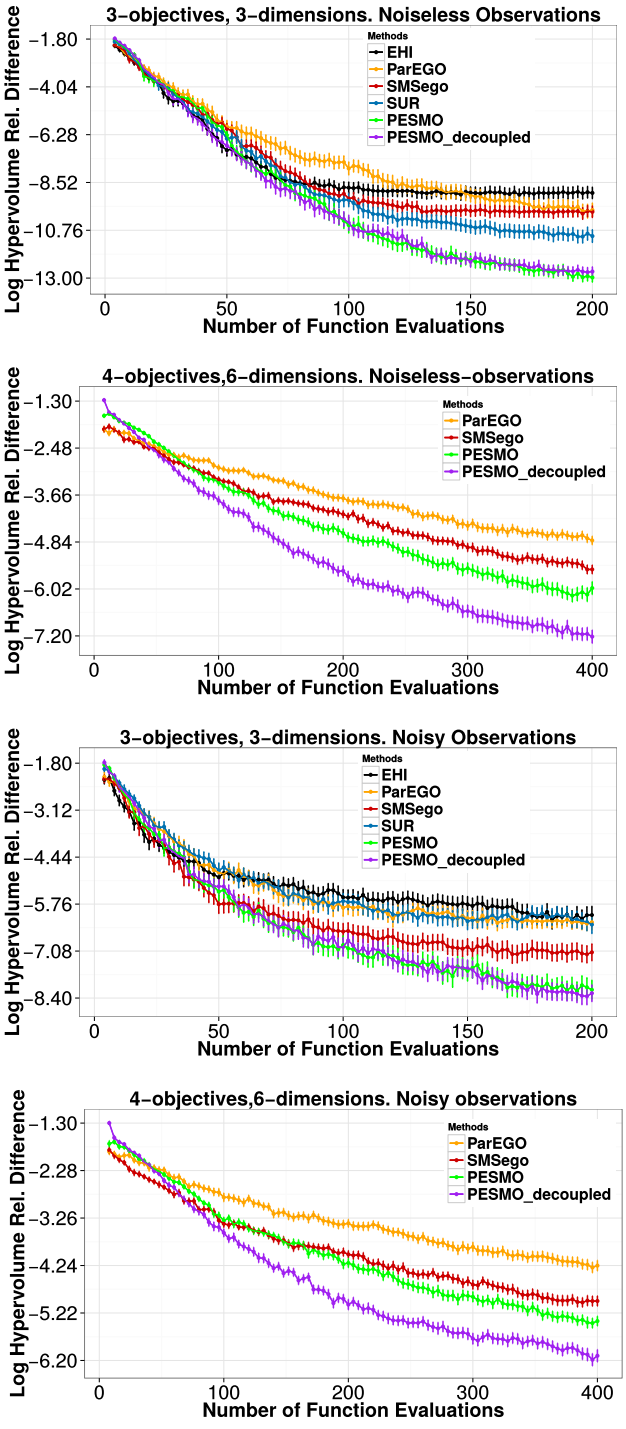


Fig. 4.3 Average log relative difference between the hyper-volume of the recommendation and the maximum hyper-volume for each number of evaluations made. We consider noiseless and noisy observations. The problem considered has 2 objectives and 3 dimensions. Similar results for a problem with 4 objectives and 6 dimensions. We do not compare results with EHI and SUR because they are infeasible due to their exponential cost with the number of objectives.

Table 4.1 Avg. time in seconds doing calculations per iteration.

PESMO	**PESMO$_{\text{dec}}$**	**ParEGO**	**SMSego**	**EHI**	**SUR**
33±1.0	52±2.5	11±0.2	16±1.3	405±115	623±59

Table 4.1 shows the average time in seconds required by each method to determine the next evaluation. The fastest method is ParEGO followed by SMSego and PESMO. The decoupled version of PESMO, PESMO$_{\text{dec}}$, takes more time because it has to optimise $\alpha_1(\cdot)$ and $\alpha_2(\cdot)$. The slowest methods are EHI and SUR; most of their cost is in the last iterations, in which the Pareto set size, $|\tilde{\mathcal{X}}^\star|$, is large due to non-dominated observations. The cost of evaluating the acquisition function in EHI and SUR is $\mathcal{O}((|\tilde{\mathcal{X}}^\star|+1)^L)$, leading to expensive optimisation via L-BFGS. In PESMO the cost of evaluating $\alpha(\cdot)$ is $\mathcal{O}(L|\mathcal{X}^\star_{(s)}|^3)$ because L linear systems are solved. These computations are faster because they are performed using the open-BLAS library, which is optimised for each processor. The acquisition function of EHI and SUR does not involve solving linear systems and hence these methods cannot use open-BLAS. Note that we also keep fixed $|\mathcal{X}^\star_{(s)}| = 50$ in PESMO.

We have carried out additional synthetic experiments with 4 objectives on a 6-dimensional input space. In this case, EHI and SUR become infeasible, so we do not compare results with them. Again, we sample the objectives from the GP prior. Figure 4.3 shows, as a function of the evaluations made, the average performance of each method. The best method is PESMO, and in this case, the decoupled evaluation performs significantly better. This improvement is because in the decoupled setting, PESMO identifies the most difficult objectives and evaluates them more times. In particular, because there are 4 objectives it is likely that some objectives are more difficult than others just by chance. Figure 4.4 illustrates this behaviour for a representative case in which the first two objectives are non-linear (difficult) and the last two objectives are linear (easy). We note that the decoupled version of PESMO evaluates the first two objectives almost three times more.

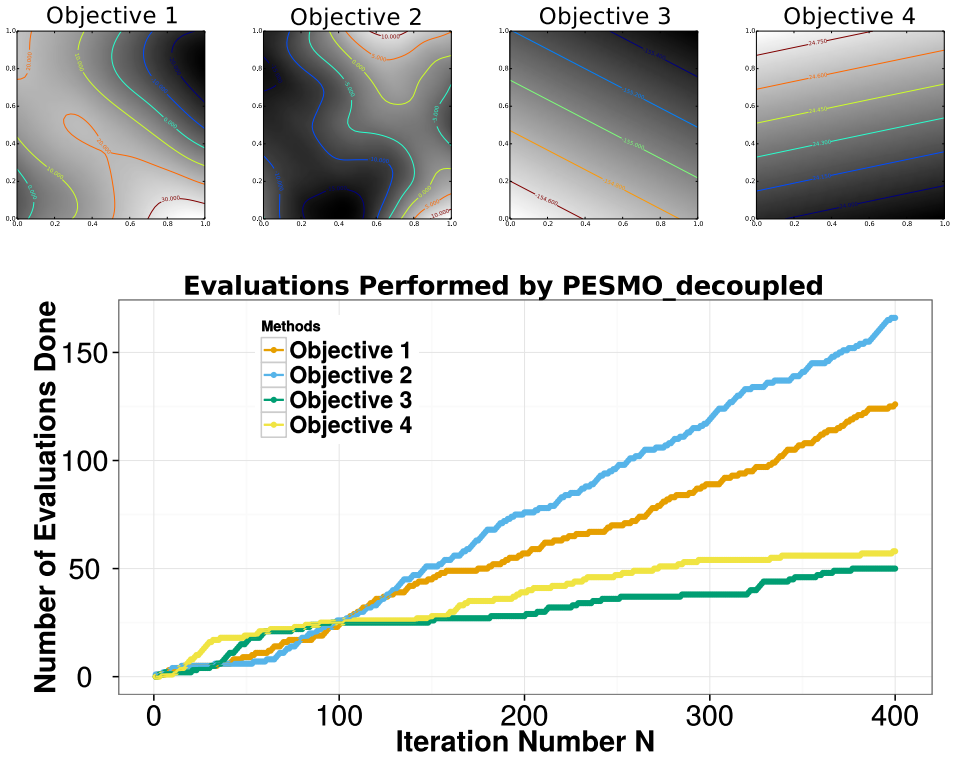


Fig. 4.4 **Top.** Contour curves of 4 illustrative objectives on 6 dimensions obtained by changing the first two dimensions in input space while keeping the other 4 fixed to zero. The first 2 objectives are non-linear while the 2 last objectives are linear. **Bottom** Number of evaluations of each objective done by $\text{PESMO}_{\text{decoupled}}$ as a function of the iterations performed N. Best seen in colour.

4.5.3 Finding a Fast and Accurate Neural Network

We consider the MNIST dataset [LeCun et al., 1998] and evaluate each method on the task of finding a neural network with low prediction error and small prediction time. These are conflicting objectives because reducing the prediction error will involve larger networks which will take longer at test time. We consider feed-forward networks with ReLus at the hidden layers and a soft-max output layer. The networks are coded in Keras (http://github.com/fchollet/keras) and they are trained using Adam [Kingma and Ba, 2014] with a minibatch size of $4,000$ instances during 150 epochs. The adjustable parameters are: the number of hidden units per layer

(between 50 and 300), the number of layers (between 1 and 3), the learning rate, the amount of dropout, and the level of ℓ_1 and ℓ_2 regularisation. The prediction error is measured on a set of $10,000$ instances extracted from the training set. The rest of the training data, *i.e.*, $50,000$ instances, is used for training. We consider a logit transformation of the prediction error because the error rates are very small. The prediction time is measured as the average time required for doing $10,000$ predictions. We compute the logarithm of the ratio between the prediction time of the network and the prediction time of the fastest network, (*i.e.*, a single hidden layer and 50 units). When measuring the prediction time we do not train the network and consider random weights (in Spearmint the time objective is also set to ignore irrelevant parameters). Thus, the problem is suited for a decoupled evaluation because both objectives can be evaluated separately. We run each method for a total of 200 evaluations of the objectives and report results after 100 and 200 evaluations. Because there is no ground truth and the objectives are noisy, we re-evaluate 3 times the values associated with the recommendations made by each method (in the form of a Pareto set) and average the results. Then, we compute the Pareto front (*i.e.*, the function space values of the Pareto set) and its hyper-volume. We repeat these experiments 50 times and report the average results across repetitions.

Table 4.2 shows the hyper-volumes obtained in the experiments (the higher, the better). The best results, after 100 evaluations of the objectives, correspond to the decoupled version of PESMO, followed by SUR and by the coupled version. When 200 evaluations are done, the best method is PESMO in either setting, *i.e.*, coupled or decoupled. After PESMO, SUR gives the best results, followed by SMSego and EHI. ParEGO is the worst performing method in either setting. In summary, PESMO gives the best overall results, and its decoupled version performs much better than the other methods when the number of evaluations is small.

Figure 4.5 shows the average Pareto front obtained by each method after 100 and 200 evaluations of the objectives. The results displayed are consistent with the ones in Table 4.2. In particular, PESMO is able to find networks that are faster than the ones found by the other methods, for a similar

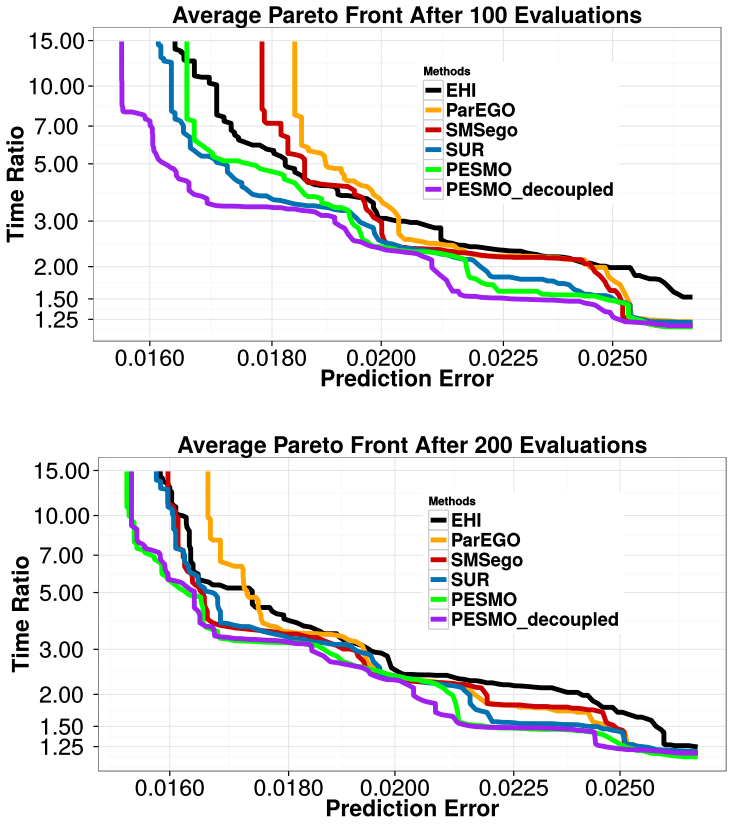


Fig. 4.5 Avg. Pareto fronts obtained by each method after 100 (left) and 200 (right) evaluations of the neural network objectives.

prediction error on the validation set. This is especially the case of PESMO when executed in a decoupled setting, after doing only 100 evaluations of the objectives. We also note that PESMO finds the most accurate networks, with almost 1.5% of prediction error in the validation set.

The good results obtained by $\text{PESMO}_{\text{decoupled}}$ are explained by Figure 4.6, which shows the average number of evaluations of each objective. More precisely, the objective that measures the prediction time is evaluated just a few times. This makes sense because it depends on only two parameters, *i.e.*, the number of layers and the number of hidden units per layer. It is hence simpler than the prediction error. $\text{PESMO}_{\text{decoupled}}$ is able to detect this and focuses on the evaluation of the prediction error. Of course, evaluating the prediction error more times is more expensive, since it involves training the

Table 4.2 Avg. hyper-volume after 100 and 200 evaluations.

# Eval.	PESMO	$\text{PESMO}_{\text{dec}}$	ParEGO	SMSego	EHI	SUR
100	66.2±.2	**67.6±.1**	62.9±1.2	65.0±.3	64.0±.9	66.6±.2
200	**67.8±.1**	**67.8±.1**	66.1±.2	67.1±.2	66.6±.2	67.2±.1

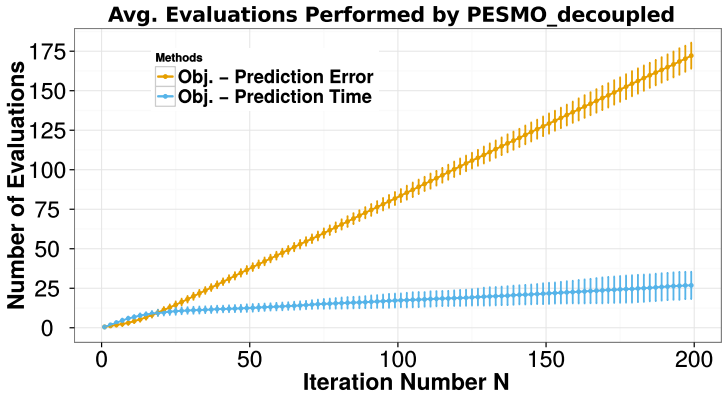


Fig. 4.6 Number of evaluations of each objective done by $\text{PESMO}_{\text{decoupled}}$, as a function of the iteration number N, in the problem of finding good neural networks.

neural network more times. Nevertheless, this shows that $\text{PESMO}_{\text{decoupled}}$ is able to successfully discriminate between easy and difficult objective functions.

4.5.4 Finding a Small and Accurate Ensemble of Decision Trees

In this section we evaluate each of the methods in the task of finding an ensemble of decision trees of small size that has low prediction error. We measure the ensemble size in terms of the sum of the total number of nodes in each of trees of the ensemble. Note that the objectives considered are conflicting because it is expected that an ensemble of small size has higher prediction error than an ensemble of larger size. The dataset considered is the *German Credit* dataset, which is extracted from the UCI repository

Table 4.3 Avg. hyper-volume after 100 and 200 evaluations of the objectives.

# Eval.	PESMO	$\text{PESMO}_{\text{dec}}$	ParEGO	SMSego	EHI	SUR
100	8.742±.006	**8.755±.009**	8.662±.019	8.719±.012	8.731±.009	8.739±.007
200	**8.764±.007**	8.758±.007	8.705±.008	8.742±.006	8.727±.008	8.756±.006

[Lichman, 2013]. This is a binary classification dataset with 1,000 instances and 9 attributes. The prediction error is measured using a 10-fold-cross validation procedure that is repeated 5 times to reduce the variance of the estimates.

Critically, to get ensembles of decision trees with good prediction properties one must encourage diversity in the ensemble [Dietterich, 2000]. In particular, if all the decision trees are equal, there is no gain from aggregating them in an ensemble. However, too much diversity can also lead to ensembles of poor prediction performance. For example, if the predictions made are completely random, one cannot obtain improved results by aggregating the individual classifiers. In consequence, we consider here several mechanisms to encourage diversity in the ensemble, and let the amount of diversity be specified in terms of adjustable parameters.

To build the ensemble we employed decision trees in which the data is split at each node, and the best split is chosen by considering each time a random set of attributes —we use the *Decision-Tree* implementation provided in the python package *scikit-learn* for this, and the number of random attributes is an adjustable parameter. This is the approach followed in Random Forest [Breiman, 2001] to generate the ensemble classifiers. Each tree is trained on a random subset of the training data of a particular size, which is another adjustable parameter. This approach is known in the literature as subbagging [Bühlmann and Yu, 2001], and has been shown to lead to classification ensembles with good prediction properties. We consider also an extra method to introduce diversity known as class-switching [Martínez-Muñoz and Suárez, 2005]. In class-switching, the labels of a random fraction of the

training data are changed to a different class. The final ensemble prediction is computed by majority voting.

In summary, the adjustable parameters are: the number of decision trees built (between 1 and $1,000$), the number of random features considered at each split in the building process of each tree (between 1 and 9), the minimum number of samples required to split a node (between 2 and 200), the fraction of randomly selected training data used to build each tree, and the fraction of training instances whose labels are changed (after doing the sub-sampling process).

Finally, we note that this setting is suited to the decoupled version of PESMO since both objectives can be evaluated separately. In particular, the total number of nodes is estimated by building only once the ensemble without leaving any data aside for validation, as opposed to the cross-validation approach used to estimate the ensemble error, which requires to build several ensembles on subsets of the data, to then estimate the prediction error on the data left out for validation.

We run each method for a total of 200 evaluations of the objectives and report results after 100 and 200 evaluations. As in the experiments with neural networks, we re-estimate three times the objectives associated to each point in the final recommendation made by each method, and average results. The goal of this averaging process is to reduce the noise in the final evaluation of the objectives, which is used to estimate the performance of each method using the hyper-volume. We repeat these experiments 50 times and report the average results across repetitions.

Table 4.3 shows the average hyper-volume of the recommendations made by each method, after 100 and 200 evaluations of the objective functions. The table also shows the corresponding error bars. In this case the observed differences among the different methods are smaller than in the experiments with neural networks. Nevertheless, we observe that the decoupled version of PESMO obtains the best results after 100 evaluations. After this, PESMO in the coupled setting performs best, closely followed by SUR. After 200

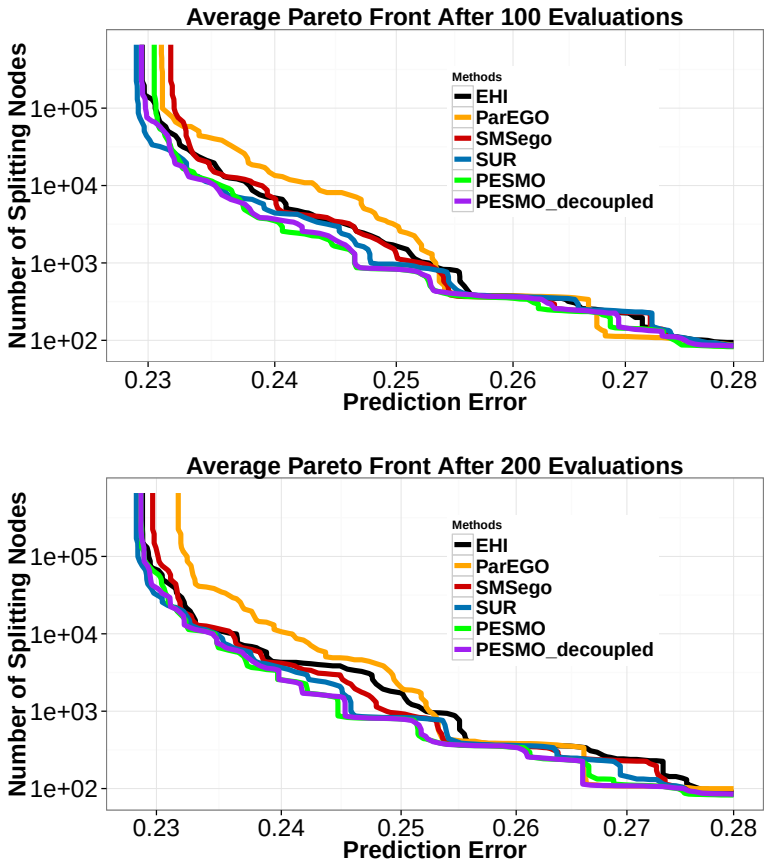


Fig. 4.7 Avg. Pareto fronts obtained by each method after 100 (top) and 200 (bottom) evaluations of the decision tree ensemble objectives.

evaluations, the best method is the coupled version of PESMO, closely followed by its decoupled version and by SUR. SMSego and EHI give worse results than these methods, in general. Finally, as in the experiments with neural networks, ParEGO is the worst performing method. In summary, the best methods are PESMO in either setting (coupled or decoupled) and SUR. All other methods perform worse. Furthermore, the decoupled version of PESMO gives slightly better results at the beginning, *i.e.*, after 100 evaluations.

Figure 4.7 shows the average Pareto front (this is simply the values in functional space associated to the Pareto set) corresponding to the recom-

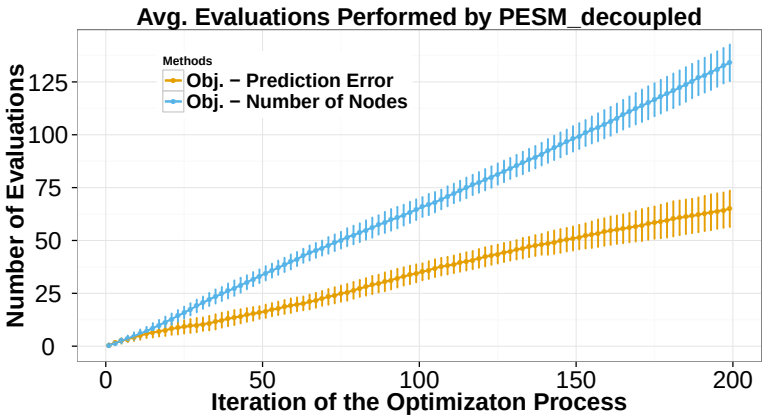


Fig. 4.8 Number of evaluations of each objective done by $\text{PESMO}_{\text{decoupled}}$, as a function of the iteration number N, in the problem of finding a good ensemble of decision trees.

mendations made by each method after 100 (top) and 200 evaluations of the objectives (bottom). We observe that PESMO finds ensembles with better properties than the ones found by EHI, SMSego and ParEGO. Namely, ensembles of smaller size for a similar or even better prediction error. The most accurate ensembles are found by SUR. Nevertheless, they have a very similar error to the most accurate ensembles found by PESMO. Finally, we note that in some cases, PESMO is able to find ensembles of intermediate size with better prediction error than the ones found by SUR.

Figure 4.8 shows the average number of times that the decoupled version of PESMO evaluates each objective. We observe that in this case the objective that measures the number of nodes in the ensemble is evaluated more times. However, the difference between the number of evaluations of each objective is smaller than the difference observed in the case of the experiments with neural networks. Namely, 135 evaluations of one objective versus 65 evaluations of the other, in this case, compared to 175 evaluations versus 25, in the case of the experiments with neural networks. This may explain why in this case the differences between the coupled and the decoupled version of PESMO are not as big as in the previous experiments.

4.6 Conclusions

We have described PESMO, a method for multi-objective Bayesian optimisation. At each iteration, PESMO evaluates the objective functions at the input location that is most expected to reduce the entropy of posterior estimate of the Pareto set. Several synthetic experiments show that PESMO has better performance than other methods from the literature. That is, PESMO obtains better recommendations with a smaller number of evaluations, both in the case of noiseless and noisy observations. Furthermore, the acquisition function of PESMO can be understood as a sum of L individual acquisition functions, one per each of the L objectives. This allows for a *decoupled* evaluation scenario, in which the most promising objective is identified by maximising the individual acquisition functions. When run in a decoupled evaluation setting, PESMO is able to identify the most difficult objectives and, by focusing on their evaluation, it provides better results. This behaviour of PESMO has been illustrated on a multi-objective optimisation problem that consists of finding an accurate and fast neural network. Finally, the computational cost of PESMO is small. In particular, it scales linearly with the number of objectives L. Other methods have an exponential cost with respect to L which makes them infeasible for more than 3 objectives.

Chapter 5

Pareto Frontier Learning with Correlated Expensive Objectives

In the previous chapter, we introduced the notion of multi-objective optimisation under a Bayesian theoretic framework, and devised an entropy based acquisition function with subsequent approximations in order to direct where the objectives should be evaluated next, to find a Pareto efficient set. A key choice that must be made in the Bayesian optimisation framework is that of the prior over objective functions.

All Gaussian process based multi-objective optimisation methods in the literature until now, to the best of our knowledge, assume *independent* Gaussian process priors on each of the objectives. However, in practice, it is highly likely that the various objectives we wish to optimise would in fact be *correlated*. For example, suppose we are trying to design a car so as to minimise the total expense whilst maximising the top speed of the car. It is reasonable to assume that the materials and processes required to maximise the top speed will also be the ones that lead to the highest build expense, as they will be in high demand from most car manufacturers. Therefore, it would be appropriate for our model of the objective functions to be able to capture the likely positive correlation between car expense and top speed. It would be reasonable to believe that correct modelling of this correlation should

lead to more judicious suggestions as to where to evaluate the functions next.

We use correlated Gaussian process priors to model the objectives, and develop an analytic approximation to the expected increase in hyper-volume acquisition function. This research was conducted in parallel to the research in the previous chapter involving predictive entropy search. We hope to combine the ideas in future work. Results from experiments on several benchmark and real world multi-objective optimisation tasks suggest that our approach of incorporating correlations is indeed highly beneficial to performance. The majority of this research was conducted with Zoubin Ghahramani and published at ICML [Shah and Ghahramani, 2016]. This work was carried out in parallel to the multi-objective predictive entropy search work from the previous chapter, where independent Gaussian process models are used for each objective, and a different acquisition function is used to decide where to evaluate the objectives next.

5.1 Pareto Hyper-volume Based Multi-objective Optimisation

The problem formulation in this chapter is the same as that of the last chapter, namely that our aim is to jointly maximise $L \geq 2$ bounded objectives, $f_l : \mathcal{X} \to \mathbb{R}$ for $l = 1, ..., L$, and find a set of Pareto efficient points. Recall that for a set of distinct points $\mathcal{Y} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_n\}$, the subset of Pareto efficient points, $\mathcal{P}(\mathcal{Y}) \subseteq \mathcal{Y}$, is defined as $\mathcal{P}(\mathcal{Y}) = \Big\{\boldsymbol{y}_i \in \mathcal{Y} : \boldsymbol{y}_j \not\succeq \boldsymbol{y}_i \forall \boldsymbol{y}_j \in \mathcal{Y} \backslash \{\boldsymbol{y}_i\}\Big\}$.

5.1.1 Definition of Pareto Hyper-volume

The *Pareto hyper-volume* is an appropriate measure of the quality of a set of Pareto efficient points [Zitzler and Thiele, 1999a]. Define a reference point, $\boldsymbol{v}_{\text{ref}} \in \mathbb{R}^L$, which is dominated by each element of $\mathcal{P}(\mathcal{Y})$ i.e. $\boldsymbol{u} \succeq \boldsymbol{v}_{\text{ref}}$ for each $\boldsymbol{u} \in \mathcal{P}(\mathcal{Y})$. The Pareto hyper-volume of $\mathcal{P}(\mathcal{Y})$ with respect to $\boldsymbol{v}_{\text{ref}}$ is

$$\text{Vol}_{\boldsymbol{v}_{\text{ref}}}\Big(\mathcal{P}(\mathcal{Y})\Big) = \int_{\mathbb{R}^L} \mathbb{I}\Big[\boldsymbol{y}' \succeq \boldsymbol{v}_{\text{ref}}\Big]\Bigg[1 - \prod_{\boldsymbol{u} \in \mathcal{P}(\mathcal{Y})} \mathbb{I}\Big[\boldsymbol{u} \not\succeq \boldsymbol{y}'\Big]\Bigg]\, d\boldsymbol{y}' \tag{5.1}$$

where $\mathbb{I}(.)$ is the indicator function, which outputs 1 if its argument is true and 0 otherwise. $\text{Vol}_{\boldsymbol{v}_{\text{ref}}}\Big(\mathcal{P}(\mathcal{Y})\Big)$ measures the volume of points in $\mathbb{R}^L$ which dominate $\boldsymbol{v}_{\text{ref}}$ but are dominated by at least one element of the Pareto set, $\mathcal{P}(\mathcal{Y})$. The shaded region of Figure 5.1(a) illustrates this volume. The Pareto hyper-volume is a monotone function since $\text{Vol}_{\boldsymbol{v}_{\text{ref}}}\Big(\mathcal{P}\Big(\mathcal{Y} \cup \{\boldsymbol{y}\}\Big)\Big) \geq \text{Vol}_{\boldsymbol{v}_{\text{ref}}}\Big(\mathcal{P}(\mathcal{Y})\Big)$.

The more dominant the set of Pareto points, the larger the Pareto hyper-volume. Conversely, a marginally dominant set of Pareto points will have a small Pareto hyper-volume. This makes the Pareto hyper-volume a reasonable measure of how "good" a proposed set of Pareto efficient points is.

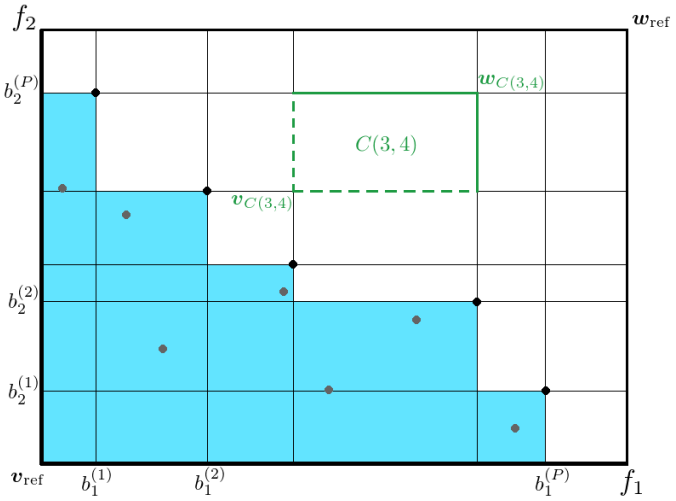
Note that the units of the hyper-volume measure is the product of the units of each of the objectives, f_l. Whilst the scale of units does not affect performance in most commonly used single objective Bayesian optimisation algorithms, the relative scales of objectives $f_1, ..., f_L$ will affect the hyper-volume measure that we propose here.

5.1.2 Expected Improvement in Pareto Hyper-volume

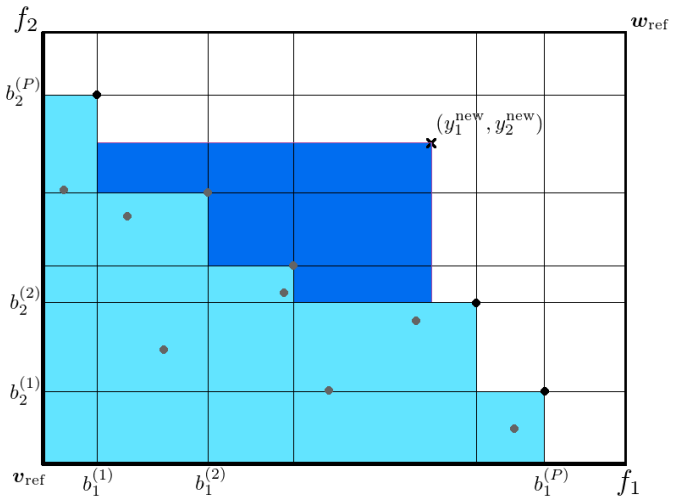
Analogous to the single objective case, we can formulate a multi objective Bayesian optimisation problem as maximising a future reward, $r_T = \Big[\text{Vol}_{\boldsymbol{v}_{\text{ref}}}\Big(\mathcal{P}(\tilde{\mathcal{Y}}_T)\Big) - \text{Vol}_{\boldsymbol{v}_{\text{ref}}}\Big(\mathcal{P}(\mathcal{Y}^*)\Big)\Big]$, where $\mathcal{Y}^*$ is the true Pareto frontier and $\tilde{\mathcal{Y}}_T$ is the suggested Pareto frontier after T evaluations of each of the objectives.

Computing the expected effect of a decision made at time step $t = 1$ on a regret at time $T \gg 1$ is computationally infeasible in the Bayesian optimisation setting [Osborne et al., 2009]. A common greedy, but computationally feasible alternative is to repeatedly maximise the expected one step ahead reward, examples of which include expected improvement, probability of improvement [Kushner, 1964], upper confidence bound [Cox and John, 1992] and entropy search [Hennig and Schuler, 2012].

Emmerich [2005] introduced the idea of *expected improvement in Pareto*



(a) Pareto hyper-volume



(b) Increase in Pareto hyper-volume

Fig. 5.1 Two objective example illustrating Pareto optimality, Pareto hyper-volume and our notation. (a) Observations $(f_1(x_i), f_2(x_i))$ for $i = 1, ..., 12$ shown by dots, with the dark dots representing the set of Pareto efficient observations. Grey dots each have at least one dark dot to the top-right of it i.e. they are dominated. f_1 and f_2 values are shown on the x-axis and y-axis respectively. The Pareto efficient points induce a grid cell partitioning of the relevant region which has bottom left corner, $\boldsymbol{v}_{\text{ref}}$, and top right corner, $\boldsymbol{w}_{\text{ref}}$. Cell $C(3,4)$ defines the cuboid from $\boldsymbol{v}_{C(3,4)} \equiv \left(b_1^{(3)}, b_2^{(4)}\right)$ to $\boldsymbol{w}_{C(3,4)} \equiv \left(b_1^{(4)}, b_2^{(5)}\right)$. The volume of the shaded region represents the Pareto hyper-volume with respect to reference point $\boldsymbol{v}_{\text{ref}}$. (b) Change in Pareto frontier from a new observation at x^{new} with value $\left(f_1(x^{\text{new}}), f_2(x^{\text{new}})\right) = \left(y_1^{\text{new}}, y_2^{\text{new}}\right)$. The new observation dominates 2 previously Pareto optimal points. The increase in Pareto hyper-volume is equal to the volume of the darker shaded region. The darker shaded region is the sum of cuboidal volumes over the previously non-dominated cells in $\mathcal{C}_{\text{nd}}$.

hyper-volume, defined as

$$\text{EIPV}(\boldsymbol{x}_{t+1}|\mathcal{D}) = \mathbb{E}_{p(\boldsymbol{y}(\boldsymbol{x}_{t+1})|\mathcal{D})}\left[\text{Vol}\Big(\mathcal{P}\big(\mathcal{Y} \cup \{\boldsymbol{y}(\boldsymbol{x}_{t+1})\}\big)\Big) - \text{Vol}\Big(\mathcal{P}\big(\mathcal{Y}\big)\Big)\right]$$

where $\boldsymbol{y}_s = [f_1(\boldsymbol{x}_s), ..., f_L(\boldsymbol{x}_s)]^\top$, $\mathcal{Y} = \{\boldsymbol{y}_s\}_{s=1}^t$, $\mathcal{D} = \{\boldsymbol{x}_s, \boldsymbol{y}_s\}_{s=1}^t$ and $\boldsymbol{v}_{\text{ref}}$ is dropped for convenience.

5.1.3 Grid Partitioning of the Output Space

Given that each f_l is bounded above, we choose a reference point $\boldsymbol{w}_{\text{ref}}$, such that $\boldsymbol{w}_{\text{ref}} \succeq [f_1(\boldsymbol{x}), ..., f_L(\boldsymbol{x})]^\top$ for any $\boldsymbol{x} \in \mathcal{X}$ (it is possible to set $w_{\text{ref},l} = \infty$). The cuboidal set of interest becomes $\mathbb{A} \equiv \{\boldsymbol{y} \in \mathbb{R}^L : \boldsymbol{w}_{\text{ref}} \succeq \boldsymbol{y} \succeq \boldsymbol{v}_{\text{ref}}\}$. Let the Pareto efficient subset be $\mathcal{P}(\mathcal{Y}) = \{\boldsymbol{u}_1, ..., \boldsymbol{u}_P\}$ for $1 \leq P \leq t$, and set $\boldsymbol{u}_0 = \boldsymbol{v}_{\text{ref}}$ and $\boldsymbol{u}_{P+1} = \boldsymbol{w}_{\text{ref}}$. Now let $b_j^{(0)} \leq ... \leq b_j^{(P+1)}$ be the sorted list of j^{th} coordinates of $\boldsymbol{u}_0, ..., \boldsymbol{u}_{P+1}$. The grid coordinates $b_j^{(p)}$ induce a cuboidal partitioning of $\mathbb{A}$. Specifically, for each $(i_1, ..., i_L) \in \{0, ..., P\}^L$, we define the grid cell $C(i_1, ..., i_L)$ as the cuboid $\left(b_1^{(i_1)}, b_1^{(i_1+1)}\right] \times \left(b_2^{(i_2)}, b_2^{(i_2+1)}\right] \times ... \times \left(b_L^{(i_L)}, b_L^{(i_L+1)}\right]$, then grid cells are disjoint and their union equals $\mathbb{A}$. Analogous to the definitions of $\boldsymbol{v}_{\text{ref}}$ and $\boldsymbol{w}_{\text{ref}}$ with respect to $\mathbb{A}$, we define $\boldsymbol{v}_{C(i_1,...,i_L)} = \left(b_1^{(i_1)}, ..., b_L^{(i_L)}\right)^\top$ and $\boldsymbol{w}_{C(i_1,...,i_L)} = \left(b_1^{(i_1+1)}, ..., b_L^{(i_L+1)}\right)^\top$. Hence for any $\boldsymbol{y} \in C(i_1, ..., i_L)$, $\boldsymbol{w}_{C(i_1,...,i_L)} \succeq \boldsymbol{y} \succeq \boldsymbol{v}_{C(i_1,...,i_L)}$. The set of grid cells is defined as $\mathcal{C} \equiv \left\{C(i_1, ..., i_L) : (i_1, ..., i_L) \in \{0, ..., P\}^L\right\}$. An example of this grid partitioning with $L = 2$ objectives is shown in Figure 5.1(a).

Note that there are two key types of grid cells: those whose points are dominated by at least one member of the Pareto efficient set and those whose points are not dominated by any member of the Pareto efficient set (shaded differently in Figure 5.1(b)). Subsequent new observations amongst the non-dominated cells change the Pareto frontier, whilst observations in the dominated cells do not. We define the set of non-dominated cells as $\mathcal{C}_{\text{nd}} \equiv \{C \in \mathcal{C} : \forall \boldsymbol{y} \in C, \boldsymbol{u} \in \mathcal{P}(\mathcal{Y}), \boldsymbol{u} \not\succeq \boldsymbol{y}\}$.

With the notation, definitions and illustrations developed, the increase in Pareto volume from a new observation, $\boldsymbol{y}^{\text{new}}$, is given by $\sum_{C \in \mathcal{C}_{\text{nd}}} \text{Vol}_{\boldsymbol{v}_C}\left(\{\boldsymbol{y}^{\text{new}}\}\right)$, because

$$
\begin{aligned}
&\mathrm{Vol}_{\boldsymbol{v}_{\mathrm{ref}}}\Big(\mathcal{P}\big(\mathcal{Y}\cup\boldsymbol{y}^{\mathrm{new}}\big)\Big)-\mathrm{Vol}_{\boldsymbol{v}_{\mathrm{ref}}}\Big(\mathcal{P}\big(\mathcal{Y}\big)\Big)\\
&\qquad=\int_{\mathbb{R}^L}\mathbb{I}\Big[\boldsymbol{o}\succeq\boldsymbol{v}_{\mathrm{ref}}\Big]\left[\prod_{\boldsymbol{u}\in\mathcal{P}(\mathcal{Y})}\mathbb{I}\Big[\boldsymbol{u}\not\succeq\boldsymbol{o}\Big]-\prod_{\boldsymbol{u}\in\mathcal{P}(\mathcal{Y}\cup\boldsymbol{y}^{\mathrm{new}})}\mathbb{I}\Big[\boldsymbol{u}\not\succeq\boldsymbol{o}\Big]\right]d\boldsymbol{o}\\
&\qquad=\sum_{C\in\mathcal{C}}\int_{\mathbb{R}^L\in C}\left[\prod_{\boldsymbol{u}\in\mathcal{P}(\mathcal{Y})}\mathbb{I}\Big[\boldsymbol{u}\not\succeq\boldsymbol{o}\Big]-\prod_{\boldsymbol{u}\in\mathcal{P}(\mathcal{Y}\cup\boldsymbol{y}^{\mathrm{new}})}\mathbb{I}\Big[\boldsymbol{u}\not\succeq\boldsymbol{o}\Big]\right]d\boldsymbol{o}\\
&\qquad=\sum_{C\in\mathcal{C}_{\mathrm{nd}}}\int_{\mathbb{R}^L\in C}\left[\prod_{\boldsymbol{u}\in\mathcal{P}(\mathcal{Y})}\mathbb{I}\Big[\boldsymbol{u}\not\succeq\boldsymbol{o}\Big]-\prod_{\boldsymbol{u}\in\mathcal{P}(\mathcal{Y}\cup\boldsymbol{y}^{\mathrm{new}})}\mathbb{I}\Big[\boldsymbol{u}\not\succeq\boldsymbol{o}\Big]\right]d\boldsymbol{o}\\
&\qquad=\sum_{C\in\mathcal{C}_{\mathrm{nd}}}\int_{\mathbb{R}^L\in C}1-\mathbb{I}\Big[\boldsymbol{y}^{\mathrm{new}}\not\succeq\boldsymbol{o}\Big]\,d\boldsymbol{o}\\
&\qquad=\sum_{C\in\mathcal{C}_{\mathrm{nd}}}\mathrm{Vol}_{\boldsymbol{v}_C}\Big(\{\boldsymbol{y}^{\mathrm{new}}\}\Big).
\end{aligned}
\tag{5.2}
$$

The first equality comes from the definition of the Pareto volume. The second equality partitions the domain of the integral. The third equality comes from the fact that the integrand is 0 for every $C\notin\mathcal{C}_{\mathrm{nd}}$. The fourth equality comes from the fact that for $\boldsymbol{o}\in C\in\mathcal{C}_{\mathrm{nd}}$, $\mathbb{I}\Big[\boldsymbol{u}\not\succeq\boldsymbol{o}\Big]=1$ for each $\boldsymbol{u}\in\mathcal{P}(\mathcal{Y})\cup\mathcal{P}(\mathcal{Y}\cup\boldsymbol{y}^{\mathrm{new}})\backslash\{\boldsymbol{y}^{\mathrm{new}}\}$, by definition of $\mathcal{C}_{\mathrm{nd}}$. The final equality comes from the definition of hyper-volume.

This is the volume of points which were previously non-dominated, but are rendered dominated by $\boldsymbol{y}^{\mathrm{new}}$. Consequently, the expected increase in Pareto volume is

$$
\begin{aligned}
\mathrm{EIPV}(\boldsymbol{x}|\mathcal{D})&=\sum_{C\in\mathcal{C}_{\mathrm{nd}}}\int_{C}\mathrm{Vol}_{\boldsymbol{v}_C}\Big(\{\boldsymbol{y}\}\Big)p(\boldsymbol{y}|\mathcal{D})\,d\boldsymbol{y}\\
&=\sum_{C\in\mathcal{C}_{\mathrm{nd}}}\int_{\boldsymbol{v}_C}^{\boldsymbol{w}_C}\prod_{l=1}^{L}\Big(y_l-v_{C,l}\Big)p(\boldsymbol{y}|\mathcal{D})\,d\boldsymbol{y}.
\end{aligned}
\tag{5.3}
$$

We have developed an acquisition function to decide where multiple objectives should be evaluated next in pursuit of finding a Pareto frontier. Next, we shall discuss various Gaussian process based measures for $p(\boldsymbol{y}|\mathcal{D})$.

5.2 Pareto Learning with Gaussian Processes

Our setting is one in which evaluating objectives $f_1, ..., f_L$ is expensive and we therefore would like to learn as much as possible per set of evaluations. Modelling objectives accurately and quantifying uncertainty about predictions are both key to deciding where we should evaluate next. Gaussian processes are ideal for modelling the objectives, as they are non-parametric, provide uncertainty estimates about function values and often permit analytically tractable inference. We review how independent GP models on each f_l lead to an analytic expression for EIPV, and introduce a novel analytic approximate of EIPV when we model the f_l as correlated.

5.2.1 EIPV for Independent Gaussian Process Objectives

Emmerich [2008] show that in the case that $f_1, ..., f_L$ are independent Gaussian process draws, the expected improvement in Pareto hyper-volume can be calculated analytically. We denote the EIPV under independent GP objectives, as IEIPV and compute it below,

$$\begin{aligned}
\text{IEIPV}(\boldsymbol{x}|\mathcal{D}) &= \sum_{C\in\mathcal{C}_{\text{nd}}} \int_{\boldsymbol{v}_C}^{\boldsymbol{w}_C} \prod_{l=1}^{L} \Big(y_l - v_{C,l}\Big) p(\boldsymbol{y}|\mathcal{D})\, d\boldsymbol{y} \\
&= \sum_{C\in\mathcal{C}_{\text{nd}}} \prod_{l=1}^{L} \int_{v_{C,l}}^{w_{C,l}} \Big(y_l - v_{C,l}\Big) \phi\Big(\frac{y_l - \mu_l}{\sigma_l}\Big)\, dy_l \\
&= \sum_{C\in\mathcal{C}_{\text{nd}}} \prod_{l=1}^{L} \sigma_l^2 \Big[\Big(\phi(\beta_{C,l}) - \phi(\alpha_{C,l})\Big) + \beta_l\Big(\Phi(\beta_{C,l}) - \Phi(\alpha_{C,l})\Big)\Big],
\end{aligned}$$

where $f_l(\boldsymbol{x})|\mathcal{D} \sim \mathcal{N}\Big(f_l(\boldsymbol{x}); \mu_l, \sigma_l^2\Big)$, $\alpha_{C,l} = (w_{C,l} - \mu_l)/\sigma_l$, $\beta_{C,l} = (v_{C,l} - \mu_l)/\sigma_l$, and ϕ and Φ are the standard Gaussian p.d.f. and c.d.f. respectively. Under the assumption of independent Gaussian process objectives, not only is the EIPV analytically computable, its derivative is too. This is achieved by computing the derivatives of μ_l and σ_l with respect to the input location $\boldsymbol{x}$ and simple applications of the chain and product rule. The assumption of independence of objectives is crucial in the above derivation as it allows us to write an integral of a product as a product of univariate simple integrals.

This is a luxury which is not enjoyed when the objectives are modelled as being correlated.

5.2.2 Approximate EIPV for Correlated Gaussian Process Objectives

Denote the integral over cell, C, in equation 5.3 as

$$\Psi_C(\boldsymbol{x}) \equiv \int_{\boldsymbol{v}_C}^{\boldsymbol{w}_C} \prod_{l=1}^{L} \Big(y_l(\boldsymbol{x}) - v_{C,l}\Big) p(\boldsymbol{y}(\boldsymbol{x})|\mathcal{D}) d\boldsymbol{y}. \tag{5.4}$$

Modelling the objectives f_l as correlated Gaussian processes would lead to a posterior $\boldsymbol{f}(\boldsymbol{x})|\mathcal{D} \sim \mathcal{N}\Big(\boldsymbol{f}(\boldsymbol{x}); \boldsymbol{\mu}, \boldsymbol{\Sigma}\Big)$, where $\boldsymbol{\Sigma}$ is non-diagonal. Whilst univariate Gaussian integrals are often analytically computable, the opposite is true for general multivariate Gaussians. Under a correlated GP, the integral Ψ_C is no longer tractable. Note that

$$\Psi_C(\boldsymbol{x}) = \int_{-\infty}^{\infty} \prod_{l=1}^{L} \Big(y_l - v_{C,l}\Big) \mathbb{I}\Big[v_{C,l} < y_l \leq w_{C,l}\Big] p(\boldsymbol{y}|\mathcal{D}) d\boldsymbol{y}. \tag{5.5}$$

Define the form of the expression inside the product of equation 5.5 as $h(y) \equiv (y - v)\mathbb{I}[v < y \leq w]$. Our approach is to approximate $h(y)$ with a scaled Gaussian probability density function, $\tilde{h}(y) = z\mathcal{N}(y; \lambda, \tau^2)$, where we set z, λ, τ to moment match $h(y)$ as follows

$$\begin{aligned} z &= \int_{-\infty}^{\infty} h(y)dy = \int_{v}^{w} (y - v)dy = \frac{1}{2}(w - v)^2 \\ \lambda &= z^{-1} \int_{-\infty}^{\infty} yh(y)dy = z^{-1} \int_{v}^{w} y(y - v)dy = \frac{1}{3}(2w + v) \\ \tau^2 &= z^{-1} \int_{-\infty}^{\infty} (y - \lambda)^2 h(y)dy = z^{-1} \int_{v}^{w} (y - \lambda)^2 (y - v)dy = \frac{1}{18}(w - v)^2. \end{aligned} \tag{5.6}$$

Note that the approximation parameters z, λ, τ do not depend on the input, x. See Figure 5.2 for an example of this approximation. The nature of our approximation is similar to that made in expectation propagation [Minka, 2001]. The important difference is that expectation propagation requires approximation parameters to be learned in order to well approximate the entire integral Ψ_C, whilst our approach simply aims to well approximate

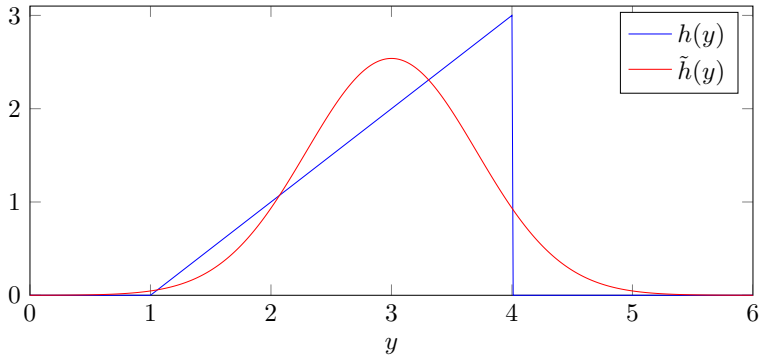


Fig. 5.2 An example of matching the true integrand factor, $h(y)$, with a scaled Gaussian, $\tilde{h}(y)$, by matching the first 3 moments.

the integrand, $h(y)$. Whilst the expectation propagation approach is more appropriate for our task statistically speaking, the parameters it would learn would strongly depend on x, which makes EP computationally too expensive. EP requires relearning the approximation parameters at each new x location whilst our approximation does not require this. Incorporating our proposed approximation strategy results in the following analytic expression for an approximation to Ψ_C,

$$\begin{aligned}
\Psi_C \approx \tilde{\Psi}_C &\equiv \int_{-\infty}^{\infty} \prod_{l=1}^{L} z_{C,l} \mathcal{N}(y_l; \lambda_{C,l}, \tau_{C,l}^2) \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma})\, d\boldsymbol{y} \\
&= \prod_{l=1}^{L} z_{C,l} \int_{-\infty}^{\infty} \mathcal{N}\Big(\boldsymbol{y}; \mathrm{diag}(\boldsymbol{\lambda}_C), \mathrm{diag}(\boldsymbol{\tau}_C^2)\Big) \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma})\, d\boldsymbol{y} \\
&= \prod_{l=1}^{L} z_{C,l} \times \exp\Bigg(-\frac{1}{2}\Big(\boldsymbol{\mu}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \log\det(\boldsymbol{\Sigma})\Big) \\
&\qquad + \frac{1}{2}\Big(\boldsymbol{\nu}_C^\top \boldsymbol{\Omega}_C^{-1}\boldsymbol{\nu}_C + \log\det(\boldsymbol{\Omega}_C)\Big) \\
&\qquad - \frac{1}{2}\sum_{l=1}^{L}\left(\frac{\lambda_{C,l}^2}{\tau_{C,l}^2} + \log\left(2\pi\tau_{C,l}^2\right)\right)\Bigg),
\end{aligned} \tag{5.7}$$

where $\boldsymbol{\Omega}_C^{-1} = \boldsymbol{\Sigma}^{-1} + \mathrm{diag}\Big(\boldsymbol{\tau}_C^2\Big)^{-1}$ and $\boldsymbol{\Omega}_C^{-1}\boldsymbol{\nu}_C = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \mathrm{diag}\Big(\boldsymbol{\tau}_C^2\Big)^{-1}\boldsymbol{\lambda}_C$. The final equality comes from the fact that the product of multivariate Gaussian probability density functions leads to a scaled multivariate Gaussian probability density function, which can be integrated analytically. We denote

the approximated expected improvement in Pareto hyper-volume under the correlated objective case as $\text{CEIPV}(\boldsymbol{x}|\mathcal{D}) = \sum_{C \in \mathcal{C}_{\text{nd}}} \tilde{\Psi}_C(\boldsymbol{x})$.

Since the parameters $\boldsymbol{z}_C, \boldsymbol{\lambda}_C, \boldsymbol{\tau}_C$ do not depend on input location, x, we compute $\partial\tilde{\Psi}_C/\partial\boldsymbol{x}$ by computing $\partial\boldsymbol{\mu}/\partial\boldsymbol{x}$ and $\partial\boldsymbol{\Sigma}/\partial\boldsymbol{x}$, repeated applications of the chain and product rules, and use of matrix derivative rules.

5.2.3 Alternative Approximation Considerations

Recall that Ψ_C ((5.4)) is the quantity we wish to approximate. An intuitive approach to approximately compute this quantity would be to use the Laplace approximation [MacKay, 2003]. We show that our approach leads to a similar looking method to that of a Laplace approximation; the latter fails to be useful in the case of correlated objectives. We may write Ψ_C as

$$\Psi_C(\boldsymbol{x}) = \int_{\boldsymbol{v}_C}^{\boldsymbol{w}_C} \prod_{l=1}^{L} \Big(y_l - v_{C,l}\Big) p(\boldsymbol{y}|\mathcal{D}) d\boldsymbol{y},$$

and define $q(\boldsymbol{y}) \equiv \prod_{l=1}^{L} \Big(y_l - v_{C,l}\Big)\mathcal{N}\Big(\boldsymbol{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma}\Big)$. Suppose there exists a $\boldsymbol{y}^*$ such that $\boldsymbol{v}_C \prec \boldsymbol{y}^* \prec \boldsymbol{w}_C$ and

$$\nabla_{\boldsymbol{y}} \log q(\boldsymbol{y})\Big|_{\boldsymbol{y}=\boldsymbol{y}^*} = (\boldsymbol{y}^* - \boldsymbol{v}_C)^{-1} - \boldsymbol{\Sigma}^{-1}(\boldsymbol{y}^* - \boldsymbol{\mu}) = \boldsymbol{0},$$

where $\boldsymbol{u}^{-1}$ is a vector with i^{th} entry u_i^{-1}. We may use a Taylor expansion approximation to see that

$$\log q(\boldsymbol{y}) \approx \log q(\boldsymbol{y}^*) - \frac{1}{2}(\boldsymbol{y} - \boldsymbol{y}^*)^\top \boldsymbol{\Omega}(\boldsymbol{y} - \boldsymbol{y}^*), \tag{5.8}$$

where $\boldsymbol{\Omega} = -\nabla_{\boldsymbol{y}}\nabla_{\boldsymbol{y}} \log q(\boldsymbol{y})\Big|_{\boldsymbol{y}=\boldsymbol{y}^*} = \text{diag}\Big((\boldsymbol{y}^* - \boldsymbol{v}_C)^{-1}\Big)^2 + \boldsymbol{\Sigma}^{-1}$, where $\text{diag}(\boldsymbol{u})$ is a diagonal matrix with i^{th} diagonal entry u_i. Consequently, we can

approximate Ψ_C as follows

$$\Psi_C(\boldsymbol{x}) \approx \int_{\boldsymbol{v}_C}^{\boldsymbol{w}_C} q(\boldsymbol{y}^*) \exp\Big(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{y}^*)^\top \boldsymbol{\Omega}(\boldsymbol{y}-\boldsymbol{y}^*)\Big) d\boldsymbol{y},$$
$$= q(\boldsymbol{y}^*)(2\pi)^{\frac{L}{2}} \det(\boldsymbol{\Omega})^{-\frac{1}{2}} \int_{\boldsymbol{v}_C}^{\boldsymbol{w}_C} \mathcal{N}(\boldsymbol{y}; \boldsymbol{y}^*, \boldsymbol{\Omega}^{-1}) d\boldsymbol{y}. \tag{5.9}$$

Notice that this approximation looks similar to the one we actually use from the previous subsection. The issue with the equation above is that it is not possible to analytically compute the c.d.f. of a general multivariate Gaussian distribution. The approach we use in the subsection above avoids this issue by directly approximating the integrand $h(y)$ with a term that spans the entire real line, on which we can perform Gaussian integrals.

5.2.4 Correlated Gaussian Process Models

In this subsection we propose two correlated output Gaussian process models to use in the CEIPV framework.

Multi-task GPs

Bonilla et al. [2008] developed a framework to model correlated functions on the same input domain, $\mathcal{X}$. The idea involves a Kronecker factorisation of the covariance between $f_l(\boldsymbol{x})$ and $f_{l'}(\boldsymbol{x}')$, separating the intra-task covariance matrix from the inter-task covariance. Specifically, $\text{Cov}(f_l(\boldsymbol{x}), f_{l'}(\boldsymbol{x}')) = \text{K}_{l,l'} k(\boldsymbol{x}, \boldsymbol{x}')$, where $\mathbf{K}$ is a positive semi-definite matrix specifying inter-task similarities and k is a covariance function over $\mathcal{X}$. Suppose for inputs $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$, $\mathbf{G}$ is such that $\text{G}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, then the full covariance matrix across tasks and data points is $\mathbf{K} \otimes \mathbf{G}$, where $\otimes$ represents a Kronecker product. A benefit of this approach, is that matrix inversion costs only $O(L^3 + n^3)$ since $\Big[\mathbf{K} \otimes \mathbf{G}\Big]^{-1} = \mathbf{K}^{-1} \otimes \mathbf{G}^{-1}$. However, a potential downside is that each function is marginally identically distributed up to scaling, which may be a poor assumption for multiple objective Pareto frontier learning. Swersky et al. [2013] utilise this model for Bayesian optimisation of single objectives to transfer knowledge from previously solved similar optimisation problems.

Semi-parametric Latent Factor GP

In this framework introduced by Teh et al. [2004], the idea is to take linear combinations of non-parametric models. Specifically, we consider L independent Gaussian processes and model each objective function to be optimised as a linear combination of the Gaussian processes. This is related to factor analysis, where data is modelled as a linear combination of global features or factors. Suppose

$$\begin{aligned} g_l &\sim \mathcal{GP}\Big(0, k_l(\boldsymbol{x}, \boldsymbol{x}')\Big) \quad \text{for } l = 1, ..., L, \\ f_l(\boldsymbol{x}) &= \sum_{l'=1}^{L} \mathbf{A}_{l,l'} g_{l'}(\boldsymbol{x}), \end{aligned} \tag{5.10}$$

where k_l are parametric kernel functions over $\mathcal{X}$ and $\mathbf{A}_{l,l'}$ are real numbers. In fact, we are able to analytically marginalise out the functions g_l, and model the functions f_l as jointly Gaussian, with the following covariance structure

$$\mathrm{Cov}\Big(f_l(\boldsymbol{x}), f_{l'}(\boldsymbol{x}')\Big) = \sum_{s=1}^{L} \mathbf{A}_{l,s}\mathbf{A}_{l',s} k_s(\boldsymbol{x}, \boldsymbol{x}').$$

A consequence of marginalising out the g_l is that we now perform matrix inversion on a $LN \times LN$ sized matrix, which has complexity $O(L^3 n^3)$. In typical settings this cost would be prohibitive, but note that under a Bayesian optimisation framework this would not be a problem. We will be interested in up to 3 objective functions to optimise, and n is typically small, of the order of 100.

5.3 Experiments

In this section, we provide empirical comparisons assessing the performance of the proposed CEIPV method. We denote the CEIPV framework under the semi-parametric latent factor GPs and multi-task models, as CEIPV-SLF and CEIPV-MT respectively. Three algorithms are used for comparison: IEIPV, ParEGO [Knowles, 2006] and Random. ParEGO is a method, which, at each iteration, defines a single objective function by taking a random convex combination of the multiple objectives, and maximising the expected

improvement under the pseudo single objective to decide where to evaluate all of the objectives next. Random simply picks a point in $\mathcal{X}$ to evaluate all the objectives at next, uniformly at random. Our experiments assume the input space, $\mathcal{X}$, is a convex subset of $\mathbb{R}^D$.

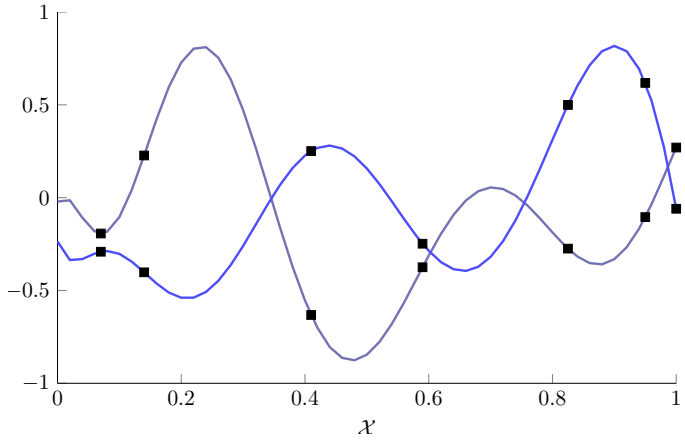
In line with Snoek et al. [2012], we choose to use ARD Matérn 5/2 kernels over the input space, defined as

$$k_{M52}(\boldsymbol{x}, \boldsymbol{x}') = \theta_0^2 \left(1 + \sqrt{5r^2} + \frac{5}{3}r^2\right) \exp\left(-\sqrt{5r^2}\right)$$
$$r^2 = \sum_{d=1}^{D} (x_d - x'_d)^2 / \theta_d^2.$$

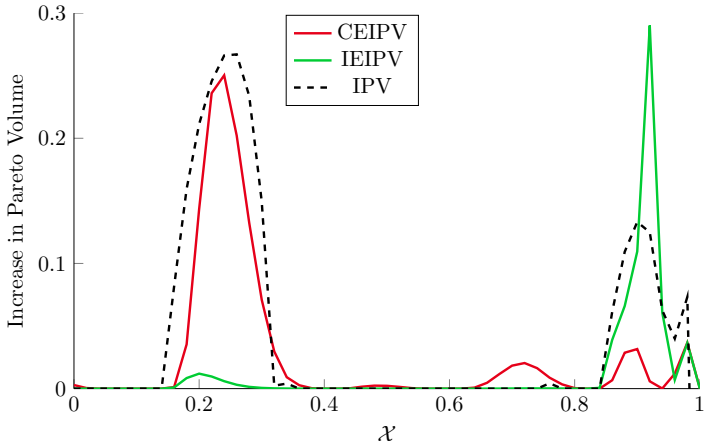
For the CEIPV algorithms, the amplitude hyperparameter, θ_0, is set to 1 to avoid over parametrisation.

In many applications, observed values are corrupted with noise. In this work, we assume each objective is observed with its own form of Gaussian noise, such that $y_l(\boldsymbol{x}) = f_l(\boldsymbol{x}) + \epsilon_l(\boldsymbol{x})$, where $\epsilon_l(\boldsymbol{x}) \sim \mathcal{N}(0, \sigma_l^2)$ independently. All of the previous derivations remain possible with the assumption of additive Gaussian noise, because a sum of Gaussians is also Gaussian distributed.

To perform a fully Bayesian treatment of the hyperparameters, we place priors over and sample them from their joint posterior given observed data using slice sampling [Neal, 2003]. As in the previous chapters, we average the acquisition functions over hyperparameter samples. Independent log-Gaussian priors are placed over $\boldsymbol{\theta}$ and $\boldsymbol{\sigma}$. In the case of CEIPV-MT, we parametrise the $L \times L$ inter-task covariance matrix $\mathbf{K}$ as $\mathbf{A}\mathbf{A}^\top$, where $\mathbf{A}$ is lower triangular. A lower triangular matrix is also used in CEIPV-SLF. For both algorithms, we place Gaussian priors on the lower triangular entries of $\mathbf{A}$.



(a) Synthetic objectives



(b) Increase in PV at $x \in \mathcal{X}$

Fig. 5.3 An illustration that modelling correlations amongst objectives is beneficial. (a) Two objectives on $\mathcal{X} = [0, 1]$ with observed function values shown by black squares. (b) Plots of CEIPV, IEIPV and IPV at new input locations. IPV is the actual increase in Pareto volume. CEIPV better matches the true IPV, as it is able to model negative correlation between objectives.

5.3.1 Empirical Illustration of the Benefit of Modelling Correlations

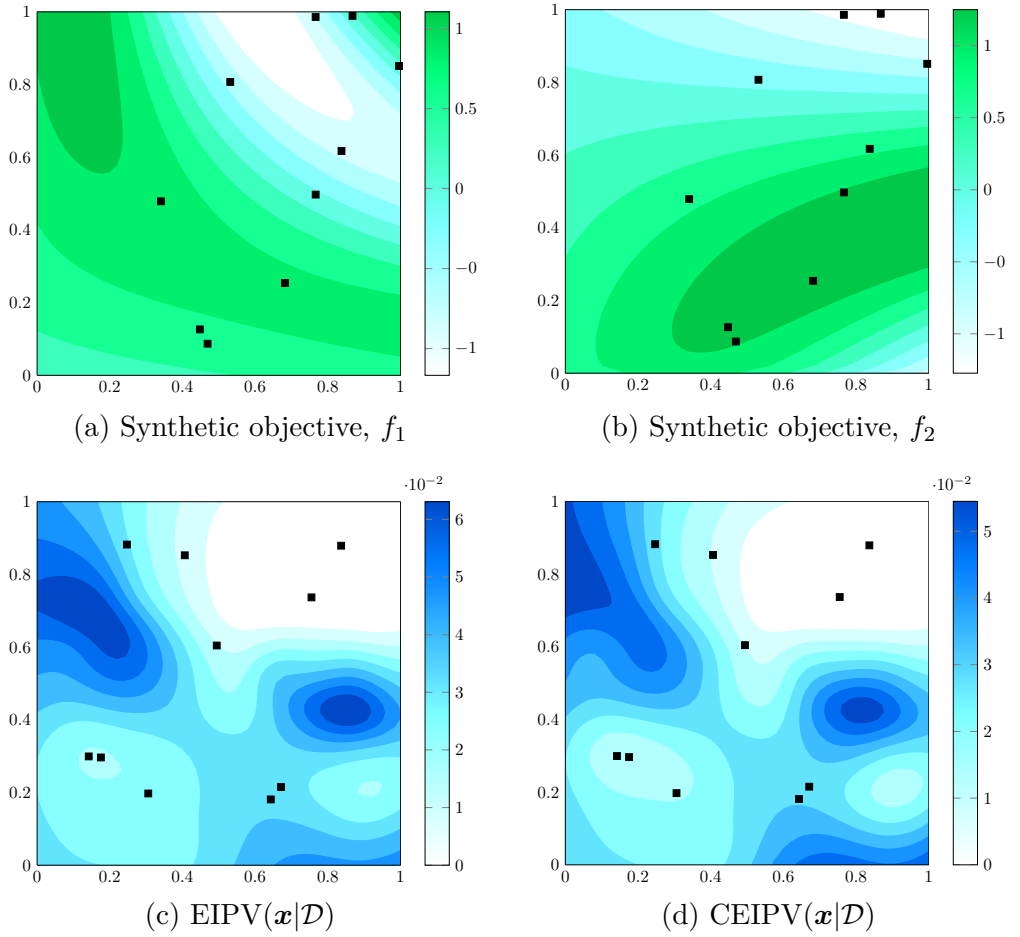
Our first experiment shows the benefit of modelling cross objective correlations, and that the CEIPV method is able to capture these correlations. For illustrative purposes, we limit the input space to $[0, 1]$, and generate

two negatively correlated objective functions, f_1 and f_2 which we jointly wish to maximise. Given noise free observations of both objectives at 7 input locations, we compared CEIPV-SLF($\boldsymbol{x}|\mathcal{D}$), IEIPV($\boldsymbol{x}|\mathcal{D}$) and the actual increase in Pareto volume from a new evaluation at $\boldsymbol{x}$ given the data and full knowledge of the objective functions, IPV($\boldsymbol{x}|\mathcal{D}, f_1, f_2$). See Figure 5.3. Notice CEIPV does a much better job of modelling IPV than IEIPV does. Whilst the independent GP method is fooled by the high function values for $\boldsymbol{x} \in [0.8, 1]$, the correlated GP model learns the strong negative correlation, and uses this to recommend that the next evaluation be in the interval $\boldsymbol{x} \in [0.2, 0.3]$.

5.3.2 Quality of the CEIPV Approximation

Once convinced that modelling correlations amongst objective functions is beneficial, we wished to assess the quality of the CEIPV approximation to the true integral, EIPV, under the the SLF model. We again consider 2 objective functions, $f_1, f_2 : [0, 1]^2 \to \mathbb{R}$ with 10 noiseless function observations (Figure 5.4(a),(b)). The objectives are constructed non-linearly from two independently drawn Gaussian processes. Under a SLF GP model, we compute an estimate of EIPV (equation 5.3) using numerical integration, and CEIPV-SLF, at new input locations $\boldsymbol{x} \in [0, 1]^2$ (Figures 5.4(c),(d)). Notice that the contours of CEIPV-SLF are on the whole very similar to those of EIPV, suggesting that the CEIPV approximation is a decent one. There is a small amount of discrepancy is in the region $[0, 0.2] \times [0.8, 1]$, where CEIPV is slightly more inclined to explore than IEPV, something we noticed in further experiments. The approximation quality appears best in regions close to observed input locations.

For the remaining experiments, we report $\mathrm{Vol}_{\boldsymbol{v}_{\mathrm{ref}}}\left(\mathcal{P}(\tilde{\mathcal{Y}}_t)\right)$ at every iteration, t. Each experiment is initialised with function evaluations at 5 input locations sampled independently and uniformly at random over the input space. To assess performance with different initial samples, we repeat each experiment 50 times with different initial input points. At each iteration, we average the acquisition function over 10 hyperparameter samples.



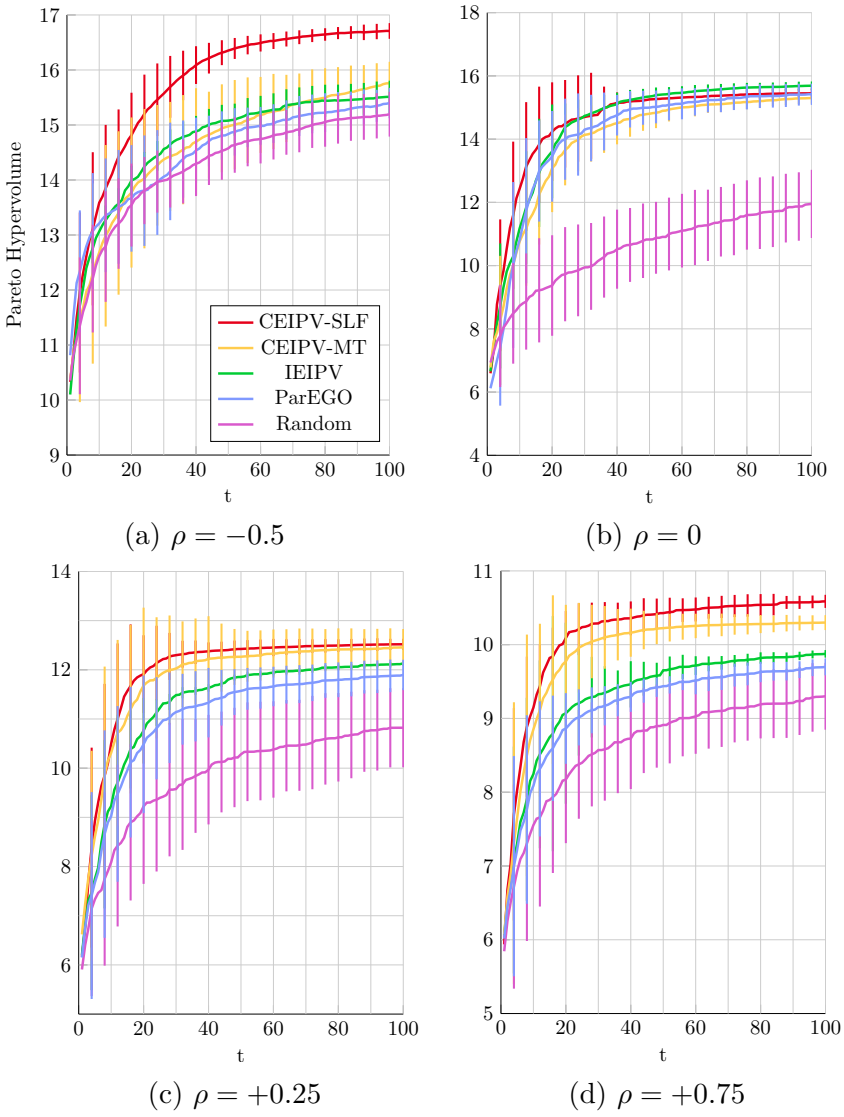
(a) Synthetic objective, f_1

(b) Synthetic objective, f_2

(c) EIPV($\boldsymbol{x}|\mathcal{D}$)

(d) CEIPV($\boldsymbol{x}|\mathcal{D}$)

Fig. 5.4 We empirically assess the quality of the CEIPV approximation to a numerical integration based estimate of EIPV on a 2 objective problem, under the Semi-parametric Latent Factor GPs model. (a), (b) Synthetic objective functions $f_1, f_2 : [0, 1]^2 \to \mathbb{R}$. Dark regions correspond to high function values and faint regions correspond to low function values. Black squares correspond to input locations of 10 function evaluations, which are identical for both functions. (c) Ground truth EIPV($\boldsymbol{x}|\mathcal{D}$), where each $\Psi_C(\boldsymbol{x})$ is computed using numerical integration over $\mathbb{R}^2$. (d) Our approximation, CEIPV($\boldsymbol{x}|\mathcal{D}$). Dark regions correspond to input locations, $\boldsymbol{x} \in \mathbb{R}$, with high utility, whilst faint regions correspond to inputs with low utility in terms of where to evaluate the objectives next.



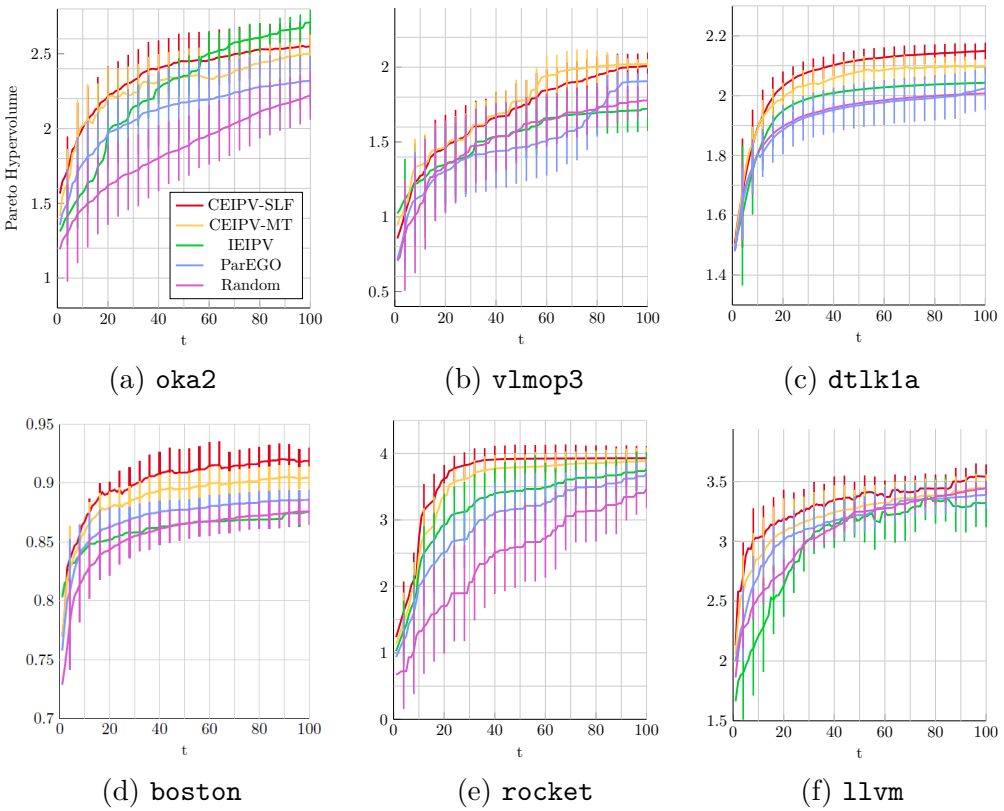
(a) $\rho = -0.5$ (b) $\rho = 0$

(c) $\rho = +0.25$ (d) $\rho = +0.75$

Fig. 5.5 Performance of various algorithms in maximising Pareto hypervolume on 2 objectives on $\mathcal{X} = [0,1]^3$, synthetically generated from the Semi-parametric Latent Factor GPs with correlations (a) $\rho = -0.5$, (b) $\rho = 0$, (c) $\rho = +0.25$ and (d) $\rho = +0.75$. Experiments for each algorithm are repeated 50 times, the mean performances plus-minus one standard deviation are plotted.

5.3.3 CEIPV Performance for Varying Levels of Actual Correlation

Next, we assess the performance of various algorithms on problems with different correlation levels. Setting the input space to $[0,1]^3$, we generate



(a) `oka2` (b) `vlmop3` (c) `dtlk1a`

(d) `boston` (e) `rocket` (f) `llvm`

Fig. 5.6 Performance of multi-objective optimisation algorithms on synthetic and real functions. Experiments for each algorithm are repeated 50 times, the mean performances plus-minus one standard deviation are plotted.

pairs of objectives, f_1, f_2, with means 2 and -2 respectively. Objectives are drawn using the SLF model with two Matérn kernels with lengthscales $\boldsymbol{\theta}^{(1)} = [0.7, 0.4, 1]$ and $\boldsymbol{\theta}^{(2)} = [0.34, 0.9, 0.5]$. The matrix $\mathbf{A}$ is set such that $\mathrm{A}_{1,1} = 1, \mathrm{A}_{2,1} = \rho$ and $\mathrm{A}_{2,2} = \sqrt{1-\rho^2}$. We generate pairs of objectives for (a) $\rho = -0.5$, (b) $\rho = 0$, (c) $\rho = +0.25$ and (d) $\rho = +0.75$. The various algorithms are run attempting to find Pareto efficient frontiers, with the results displayed in Figure 5.5. The higher the absolute value of correlation, the more the CEIPV methods outperform other methods. For $\rho = 0$ there is so statistical difference between between IEIPV and CEIPV models as we would expect. ParEGO also performs just as well for this level of correlation. Each of the 4 model based methods outperforms Random. These experiments provide strong evidence that the CEIPV are able to account for correlations amongst objectives, and that this can lead to superior performance.

5.3.4 Performance over Benchmark Multi-objective Optimisation Tasks

While Gaussian process based multi-objective optimisation has not been thoroughly explored, there is a long history of, typically model-free based, approaches to this problem, including evolutionary and genetic algorithms [Coello et al., 2002; Zitzler and Thiele, 1999b]. Several benchmark functions have been constructed for testing the efficacy of optimisation algorithms. We choose three such functions for experimentation: `oka2` [Okabe et al., 2004], `vlmop3` [Veldhuizen and Lamont, 1999] and `dtlz1a` [Deb et al., 2001]. We define the functions below.

- `oka2` is defined for $x_1 \in [-\pi, \pi]$, $x_2, x_3 \in [-5, 5]$ as

$$
\begin{aligned}
f_1(\boldsymbol{x}) &= -x_1, \\
f_2(\boldsymbol{x}) &= -1 + \frac{1}{4\pi^2}(x_1 + \pi)^2 - |x_2 - 5\cos(x_1)|^{\frac{1}{3}} - |x_3 - 5\sin(x_1)|^{\frac{1}{3}}.
\end{aligned}
$$

- `vlmop3` is defined for $\boldsymbol{x} \in [-3, 3]^2$ as

$$
\begin{aligned}
f_1(\boldsymbol{x}) &= -0.5(x_1 + x_2)^2 - \sin(x_1^2 + x_2^2), \\
f_2(\boldsymbol{x}) &= -\frac{(3x_1 - 2x_2 + 4)^2}{8} - \frac{(x_1 - x_2 + 1)^2}{27} - 15, \\
f_3(\boldsymbol{x}) &= -\frac{1}{x_1^2 + x_2^2 + 1} + 1.1\exp(-x_1^2 - x_2^2).
\end{aligned}
$$

- `dtlz1a` is defined for $\boldsymbol{x} \in [0, 1]^6$ as

$$
\begin{aligned}
f_1(\boldsymbol{x}) &= -0.5x_1(1 + g(\boldsymbol{x})), \\
f_2(\boldsymbol{x}) &= -0.5(1 - x_1)(1 + g(\boldsymbol{x})), \\
g(\boldsymbol{x}) &= 100\left[5 + \sum_{i=2}^{6}(x_i - 0.5)^2 - \cos\Big(2\pi(x_i - 0.5)\Big)\right].
\end{aligned}
$$

Results for the experiments on these functions are shown in Figure 5.6(a),(b),(c). The objectives being experimented on have complicated correlations between them, which depend on the location of the input space. Nonetheless, the CEIPV methods perform consistently strongly in all experiments. IEIPV

does well on the `oka2` task despite a slow start. We believe this was due to f_1 being simple to model, where a correlated multi-task GP may be prone to consider more complicated explanations than necessary.

5.3.5 Performance on Real World Data and Simulations

Finally we consider three real-world multi objective Pareto frontier optimisation problems. The first problem, `boston`, involves training a 2 hidden layer neural network on a random train/test split of the Boston Housing dataset [Bache and Lichman, 2013], as we did for parallel predictive entropy search. The function takes as input the weight-decay parameter, number of training iterations and size of the hidden layers. The outputs are the negative prediction error on the test set, and the negative product of the layer size and number of training iterations. The idea is to explore the trade off between (i) accuracy of prediction, and (ii) a combined measure of memory consumption and training time of the neural network. Such a trade off would be useful to explore, for example in the case of storing neural network models on mobile devices with limited memory and computational power. A manufacturer of such devices would be interested in knowing the form of the Pareto curve as this could influence final design choices.

Next we consider `rocket`, a simulation of a rocket [Hasbun, 2008] being launched from the Earth's surface, as we did for parallel predictive entropy search. The mass of fuel used, launch height and launch angle relative to the ground are inputs to the simulation. The outputs are the time taken to return to the Earth's surface, the angular distance travelled with respect to the centre of the Earth, and the absolute difference between the launch angle and the radius at the point of launch. Simulations are often used in engineering to explore what outcomes of various design choices may be for example in aerospace and automobile engineering. Nevertheless, running simulations can take days and require vast computational resources, making intelligent experiment choice crucial.

Thirdly we consider the problem of optimising compiler settings for the

LLVM compiler, using the SW-LLVM data set of [Siegmund et al., 2012]. The design space consists of 1023 different compiler settings, determined by 10 binary flags. The objectives are memory footprint and performance on a given set of software programs, compiled with the particular compiler settings. The data was very costly to obtain; evaluating the objectives on a particular compiler setting takes several hours. We denote the problem `llvm`, and set the input space to $[0, 1]^{10}$, using a rounding function to determine the binary indicators.

Results on each of the three real world tasks are shown in Figure 5.6(d),(e),(f). As before, the CEIPV algorithms which incorporate correlations between objectives tend to perform best, by a statistically significant margin in most cases. Most interestingly, the IEIPV model performs marginally worse than random selection on the `llvm` task, whilst the CEIPV methods do significantly better than random. On the `boston` problem, the CEIPV methods appear to achieve results in 10 iterations, which takes ParEGO about 100 iterations, suggesting that our approximation and correlation modelling significantly boosts the rate at which we approach Pareto optimality.

5.4 Discussion

In this paper, we argue that modelling correlations amongst objectives in multi-objective Pareto optimisation problems is important for success. To overcome the problem of intractable integrals, we devise a novel approximation which leads to an analytic and differentiable approximation to the expected increase in Pareto hypervolume acquisition function. Two forms of correlated output GP models are implemented on a variety of multi-objective problems, and seem to consistently outperform competing models which model objectives as being independent.

Whilst our empirical results suggest that modelling correlations using the approximation we derive is beneficial, more work may be required to assess where the approximation breaks down. For example, when the objectives are truly uncorrelated, CEIPV would not match IEIPV as one would de-

sire. A possibly simple way to circumvent this issue would be to define a utility function of the form $\kappa\,\mathrm{IEIPV}(x|\mathcal{D}) + (1-\kappa)\,\mathrm{CEIPV}(x|\mathcal{D})$, where $\kappa = \det(\mathrm{correl}(\boldsymbol{\Sigma}))$ and $\mathrm{correl}(\boldsymbol{\Sigma})$ is the correlation matrix induced by $\boldsymbol{\Sigma}$, such that $\mathrm{correl}(\boldsymbol{\Sigma})_{i,j} = \frac{\Sigma_{i,j}}{\sqrt{\Sigma_{i,i}\Sigma_{j,j}}}$. The determinant of the correlation matrix is 1 exactly when the objectives are uncorrelated, and it is 0 exactly when the objectives are fully correlated. Using a linear combination of the IEIPV and CEIPV in this way removes the inconsistency of the CEIPV when the objectives are truly independent.

Another small flaw in the approach we have taken in this work is in the need to prespecify $\boldsymbol{v}_{\mathrm{ref}}$ and $\boldsymbol{w}_{\mathrm{ref}}$. In theory, setting these quantities to $-\boldsymbol{\infty}$ and $+\boldsymbol{\infty}$ is feasible, however, we found that more sensible initialisations led to smoother training. In practice, we based $\boldsymbol{v}_{\mathrm{ref}}$ on the 5 initial randomly chosen input locations, and fixed it. $\boldsymbol{w}_{\mathrm{ref}}$ was chosen to dominate all of the initial function evaluations, and was adapted if an observation was made which dominated it.

There are several directions in which this work may be extended further. Further theoretical analysis is required to assess the nature of the approximation we have made. In order to reduce computational burden under the SLF model, one may consider implementing a sparse approximation to the $NL \times NL$ covariance matrix which leads to faster computation. Next, we could work on how to select a batch of points where we can evaluate next in parallel, in a multi-objective setting. Another interesting avenue would be to experiment with alternative correlated output models, such as a deep neural network.

Chapter 6

Conclusions

Optimisation is a fundamental concept to all sorts of engineering problems. More specifically, Bayesian optimisation provides a powerful framework for optimisation of objectives which are expensive to evaluate pointwise and do not provide gradient information. This thesis presents a range of contributions to Bayesian optimisation research, including studying non-parametric Bayesian priors for continuous functions, developing acquisition functions for parallel optimisation and multi-objective optimisation.

Whilst Gaussian processes are the default choice for non-parametric Bayesian function modelling, in Chapter 2 we explore a more general class of elliptical processes. Our finding was that a Student-t process is in fact a more general non-parametric prior than the Gaussian process, and is also as general as is tractably possible. Theoretically, for a fixed covariance and mean function, the Gaussian and Student-t processes lead to the same predictive means given data, however, their predictive covariance is different, as are the higher order moments. This was evident in the Bayesian optimisation application, where GPs and TPs exhibited different exploration-exploitation trade offs, especially in low data settings. Going forward, more research is required to discover and test different models for the objective function to be optimised. Random forests and Bayesian neural network models have been explored in the recent past, but little is known about the pros and cons of various models.

In Chapter 3 we introduce the notion of statistical entropy as a way to

measure the information content of a random variable. Bayesian optimisation is concerned with finding the optimiser, $\boldsymbol{x}^*$, of an unknown function. Hennig and Schuler [2012] propose the idea of choosing where to evaluate a function next to maximally reduce the entropy of $\boldsymbol{x}^*$ given the data. We extend this idea and those of Hernández-Lobato et al. [2014], to the case were we are able to query the unknown function at multiple locations at the same time in parallel. Our approach is the first to our knowledge which selects the batch of points jointly rather than greedily. A downside of our approach was that it was often difficult to determine how severely performance was being affected by the approximations we were implicitly making by the constraints we were imposing. In a few instances, the expectation propagation procedure in the inner loop of the batch Bayesian optimisation would fail to converge. We found that this was often due to conditioning on a poor value of optimiser, $\boldsymbol{x}^*$. Sampling effectively from the posterior distrbution of the optimised given observations is very difficult, and more attention should be paid to this problem.

Maintaining the theme of information gain, we discuss as approach to decide where to probe multiple objectives next in Chapter 4. Here the task is to find the set of Pareto efficient points with respect to multiple objectives. We therefore choose to evaluate functions at a point which would maximally reduce the entropy of the Pareto frontier in expectation. Similar to the problems with parallel predictive entropy search, it proved difficult to sample points on the Pareto efficient curve given observations of the functions. However, the idea of decoupling the acquisition function over the different objective functions proved fruitful and is a direction which should be pursued further. However, in both the multi-objective and batch cases, a slight flaw was in our treatment of hyperparameters, which we discussed in the corresponding chapters. A more principled way of accounting for how observations affect your information about the hyperparameters themselves is necessary.

Finally in Chapter 5, we make the assertion that in most cases where one would like to jointly optimise multiple objectives, it is likely that the objectives are inherently correlated in some way. We therefore proceed

to model objectives as being correlated, and develop an approximation to volume based acquisition function to decide where to evaluate the multiple correlated models. Whilst partitioning the output space using grid cells was a theoretically attractive approach, it led to computational difficulties and slow algorithmic performance through significant book keeping. Furthermore, the input agnostic Gaussian approximation we made to the factors in the integrand of the expected increase in Pareto hypervolume was fairly naive and simple. Despite decent empirical performance, it is necessary to spend more time testing where such an approximation could break down. Nevertheless, incoporating dependencies across objective functions in the model seemed very beneficial to performance.

References

I. Ahmad and P. E. Lin. A Nonparametric Estimation of the Entropy for Absolutely Continuous Distributions. *IEEE Trans. on Information Theory*, 22(3):372–375, 1976.

B. S. Anderson, A. W. Moore, and D. Cohn. A Nonparametric Approach to Noisy and Costly Optimization. *International Conference on Machine Learning*, 2000.

C. Archambeau and F. Bach. Multiple Gaussian Process Models. 2010.

R. Ariizumi, M. Tesch, H. Choset, and F. Matsuno. Expensive multiobjective optimization for robotics with consideration of heteroscedastic noise. In *2014 IEEE International Conference on Intelligent Robots and Systems*, pages 2230–2235, 2014.

J. Azimi, A. Fern, and X. Z. Fern. Batch Bayesian Optimization via Simulation Matching. *Neural Information Processing Systems*, 2010.

K. Bache and M. Lichman. UCI Machine Learning Repository, 2013.

D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

B. Betrò. Bayesian Methods in Global Optimization. *Journal of Global Optimization*, 1:1–14, 1991.

S. Bochner. *Lectures on Fourier Integrals*. Princeton University Press, 1959.

E. V. Bonilla, K. M. A. Chai, and C. K. I. Williams. Multi-task Gaussian Process Prediction. *Neural Information Processing Systems*, 2008.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

E. Brochu, M. Cora, and N. de Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Applications to Active User Modeling and Hierarchical Reinforcement Learning. Technical Report TR-2009-23, University of British Columbia, 2009.

P. Bühlmann and B. Yu. Analyzing Bagging. *Annals of Statistics*, 30: 927–961, 2001.

E. H. Burrows, W. K. Wong, X. Fern, F.W.R. Chaplen, and R.L. Ely. Optimization of ph and nitrogen for enhanced hydrogen production by synechocystis sp. pcc 6803 via statistical and machine learning methods. *Biotechnology Progress*, 25(4):1009–1017, 2009.

S. Cambanis, S. Huang, and G. Simons. On the theory of elliptically contoured distributions. *Journal of Multivariate Analysis*, 11:368–385, 1981.

C. A. C. Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary algorithms for solving multi-objective problems*, volume 242. Springer, 2002.

E. Contal, D. Buffoni, D. Robicquet, and N. Vayatis. Parallel Gaussian Process Optimization with Upper Confidence Bound and Pure Exploration. In *Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer Berlin Heidelberg, 2013.

P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling Wine Preferences by Data Mining from Physiochemical Properties. *Decision Support Systems, Elsevier*, 2009.

D. D. Cox and S. John. A statistical method for global optimization. *Proc. IEEE Conference on Systems*, 1992.

N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

L. Csató and M. Opper. Sparse Online Gaussian Processes. *Neural Computation*, 14:641–669, 2002.

J. P. Cunningham, P. Hennig, and S. Lacoste-Julien. Gaussian Probabilities and Expectation Propagation. *arXiv*, 2013. http://arxiv.org/abs/1111.6832.

A. P. Dawid. Spherical Matrix Distributions and a Multivariate Model. *J. R. Statistical Society B*, 1977.

A. P. Dawid. Some Matrix-Variate Distribution Theory: Notational Considerations and a Bayesian Application. *Biometrika*, 1981.

K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Test Problems for Evolutionary Multi-Objective Optimization. Technical Report 112, TIK lab, ETH, 2001.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

T. Desautels, A. Krause, and J. Burdick. Parallelizing Exploration-Exploitation Tradeoffs with Gaussian Process Bandit Optimization. *International Conference on Machine Learning*, 2012.

T. G. Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

A. Edelman and N. Raj Rao. Random Matrix Theory. *Acta Numerica*, 1: 1–65, 2005.

A Emmerich. The computation of the expected improvement in dominated hypervolume of Pareto front approximations. Technical Report LIACS TR-4-2008, Leiden University, The Netherlands, 2008.

M. Emmerich. *Single- and Multiobjective Evolutionary Design Optimization Using Gaussian Random Field Metamodels*. PhD thesis, FB Informatik, University of Dortmund, 2005.

K. T. Fang, S. Kotz, and K. W. Ng. *Symmetric Multivariate and Related Distributions*. Chapman & Hall, 1989.

R. Garnett, M. A. Osborne, and S. J. Roberts. Bayesian Optimization for Sensor Set Selection. *ACM/IEEE Intl. Conf. on Information Processing in Sensor Networks*, 2010.

M. A. Gelbart, J. Snoek, and R. P. Adams. Bayesian optimization with unknown constraints. In *Thirtieth Conference on Uncertainty in Artificial Intelligence*, 2014.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*, volume 2. Chapman & Hall/CRC Boca Raton, FL, USA, 2014.

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

J. W. Gibbs. On the equilibrium of heterogeneous substances. *American Journal of Science*, 1878.

D. Ginsbourger, J. Janusevskis, and R. Le Riche. Dealing with Asynchronicity in Parallel Gaussian Process Based Optimization. 2011. URL https://hal.archives-ouvertes.fr/hal-00507632/.

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes. 2008.

J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 148–177, 1979.

J. E. Hasbun. In *Classical Mechanics with MATLAB Applications*. Jones & Bartlett Learning, 2008.

P. Hennig and C. J. Schuler. Entropy Search for Information-Efficient Global Optimization. *JMLR*, 2012.

D. Hernández-Lobato, J. M. Hernández-Lobato, A. Shah, and R. P. Adams. Predictive Entropy Search for Multi-objective Bayesian Optimization. *International Conference on Machine Learning*, 2016.

J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. *Neural Information Processing Systems*, 2014.

T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

N. Houlsby, J. M. Hernández-Lobato, F. Huszar, and Z. Ghahramani. Collaborative Gaussian Processes for Preference Learning. *Neural Information Processing Systems*, 2012.

I. Hupkens, M. Emmerich, and A. Deutz. Faster computation of expected hypervolume improvement. 2014. arXiv:1408.7114 [cs.DS].

E. T. Jaynes. *Probability Theory, The Logic of Science*. Cambridge University Press, 2003.

D. R. Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001a.

D. R. Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001b.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4): 455–492, 1998.

D. Kelker. Distribution Theory of Spherical Distributions and a Location-Scale Parameter. *Sankhya, Ser. A,*, 1970.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. 2014. arXiv:1412.6980.

J. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10:50–66, 2006.

H. J. Kushner. A new method of locating the maximum of arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 1964.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

X. Li. A non-dominated sorting particle swarm optimizer for multiobjective optimization. In *Genetic and Evolutionary Computation — GECCO 2003*, pages 37–48. 2003.

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

D. Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, 2008.

D. J. MacKay. Information-Based Objective Functions for Active Data Selection. *Neural Computation*, 4(4):590–604, 1992.

D. J. C. MacKay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

R. Marchant and F. Ramos. Bayesian Optimisation for Intelligent Environmental Monitoring. *Neural Information Processing Systems workshop on Bayesian Optimization and Decision Making*, 2012.

C. Marsh. Introduction to Continuous Entropy. 2013. http://www.crmarsh.com/static/pdf/Charles_Marsh_Continuous_Entropy.pdf.

R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. A. Castellanos. Active Policy Learning for Robot Planning and Exploration under Uncertainty. *Robotics Science and Systems*, 2007.

G. Martínez-Muñoz and A. Suárez. Switching class labels to generate classification ensembles. *Pattern Recognition*, 38:1483–1494, 2005.

B. Matérn. *Spatial Variation*. Springer-Verlag, 1986.

W. McCulloch and W. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biology*, 5(4):115–133, 1943.

T. Minka. EP: A Quick Reference. *Technical Report*, 2008.

T. P. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Masachusetts Institute of Technology, 2001.

J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129):2, 1978.

H. Nakayama, Y. Yun, and M.Yoon. *Sequential Approximate Multiobjective Optimization Using Computational Intelligence*. Springer, 2009.

R. M. Neal. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

R. M. Neal. Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. *Technical Report No. 9702, Dept of Statistics, University of Toronto*, 1997.

R. M. Neal. Slice sampling. *Annals of Statistics*, 2003.

R. M. Neal. *Handbook of Markov chain Monte Carlo*. Chapman & Hall/CRC, 2011.

A. O'Hagan. On Curve Fitting and Optimal Design for Regression. *Journal of the Royal Statistical Society*, 40:1–42, 1978.

T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff. On test functions for evolutionary multiobjecive optimization. *Parallel Problem Solving from Nature*, 2004.

Peter Orbanz and Yee Whye Teh. Bayesian nonparametric models. In *Encyclopedia of Machine Learning*, pages 81–89. Springer, 2011.

M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian Processes for Global Optimization. *Int'l Conf on Learning and Intelligent Optimization*, 2009.

V. Picheny. Multiobjective optimization using Gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25:1265–1280, 2015.

V. Picheny, T. Wagner, and D. Ginsbourger. A Benchmark of Kriging-Based Infill Criteria for Noisy Optimization. *Structural and Multidisciplinary Optimization*, 48(3):607–626, 2013.

W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-metric selection. In *Parallel Problem Solving from Nature – PPSN X*, pages 784–794. 2008.

J. Portilla, V. Strela, M.J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Transactions on Image processing*, 12(11):1338–1351, 2003.

A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines. *Neural Information Processing Systems*, 2007.

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

C. E. Rasmussen. *Evaluation of Gaussian Processes and other Methods for Non-Linear Regression.* PhD thesis, Graduate Department of Computer Science, University of Toronto, 1996.

H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 55, 1952.

F. Rosenblatt. *Principles of Neurodynamics.* Spartan Book, 1962.

D. Rumelhart, G. Hinton, and R. Williams. Learning Representations by Back-Propagating Errors. *Nature*, 323(6088):533–536, 1986.

M. Seeger. Expectation Propagation for Exponential Families. *Technical Report, U.C. Berkeley*, 2008.

A. Shah and Z. Ghahramani. Parallel Predictive Entropy Search for Batch Global Optimisation of Expensive Objective Functions. *Neural Information Processing Systems*, 2015.

A. Shah and Z. Ghahramani. Pareto Frontier Learning with Expensive Correlated Objectives. *International Conference on Machine Learning*, 2016.

A. Shah, A. G. Wilson, and Z. Ghahramani. Student-t Processes as Alternatives to Gaussian Processes. *AISTATS*, 2014.

C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication, publisher=University of Illinois Press IL.* 1949.

J. Shekel. Test Functions for Multimodal Search Techniques. *Information Science and Systems*, 1971.

N. Siegmund, S. S. Kolesnikov, C. Kastner, S. Apel, D. Batory, M. Rosenmuller, and G. Saake. Predicting Performance via Automated Feature-Interaction Detection. *Int'l Conf. on Software Engineering*, 2012.

H. Singh, N. Misra, V. Hnizdo, A. Fedorowicz, and E. Demchuk. Nearest neighbor estimates of entropy. *American journal of mathematical and management sciences*, 23:301–321, 2003.

E. Snelson and Z. Ghahramani. Sparse Gaussian Processes using Pseudo-Inputs. *Neural Information Processing Systems*, 2006.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *Neural Information Processing Systems*, 2012.

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. *International Conference on Machine Learning*, 2010.

K. Swersky, J. Snoek, and R. P. Adams. Multi-task Bayesian Optimization. 2013.

Y. W. Teh, M. Seeger, and M. I. Jordan. Semiparametric Latent Factor Models. *Neural Information Processing Systems*, 2004.

W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25:285–294, 1933.

J. Vanhatalo, P. Jylanki, and A. Vehtari. Gaussian Process Regression with Student-t Likelihood. pages 1910–1918, 2009.

M. H. Veatch and L. M. Wein. Scheduling a make-to-stock queue: Index policies and hedging points. *Operations Research*, 44:634–647, 1996.

D. A. V. Veldhuizen and G. B. Lamont. Multiobjective Evolutionary Algorithm Test Suites. *ACM Symposium on Applied Computing*, 1999.

J. Villemonteix, E. Vazquez, and E. Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44:509–534, 2009.

M.J. Wainwright and E.P. Simoncelli. Scale Mixtures of Gaussians and the Statistics of Natural Images. *Neural Information Processing Systems*, 1999.

Z. Wang, M. Zoghi, D. Matheson, F. Hutter, and N. de Freitas. Bayesian Multi-Scale Optimistic Optimization. *AISTATS*, 2014.

E. Westervelt and J. Grizzle. *Feedback Control of Dynamic Bipedal Robot Locomotion*. Control and Automation Series. CRC PressINC, 2007.

C. Williams and M. Seeger. Using the nystrom method to seed up kernel machines. *Neural Information Processing Systems*, pages 682–688, 2001.

A. G. Wilson and R. P. Adams. Gaussian process covariance kernels for pattern discovery and extrapolation. 2013.

A. G. Wilson, E. Gilboa, A. Nehorai, and J. P. Cunningham. GPatt: Fast multidimensional pattern extrapolation with Gaussian processes. *arXiv*, 2013. URL http://arxiv.org/abs/1310.5288.

Z. Xu, F. Yan, and Y. Qi. Sparse Matrix-Variate t Process Blockmodel. 2011.

D. Yogatama and N. A. Smith. Bayesian Optimization of Text Representations. *arXiv*, 2015. URL http://arxiv.org/abs/1503.00693.

S. Yu, V. Tresp, and K. Yu. Robust Multi-Task Learning with t-Processes. 2007.

Y. Zhang and D. Y. Yeung. Multi-Task Learning using Generalized t Process. 2010.

A. Zilinskas. On the Use of Statistical Methods in Multimodal Functions for the Construction of Optimization Algorithms. In *Lecture Notes in Control and Information Sciences*. Springer-Verlag, 1980.

E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3:257–271, 1999a.

E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE trans. Evol. Comput.*, 3(4):257–271, 1999b.

M. Zuluaga, A. Krause, G. Sergent, and M. Püschel. Active learning for multi-objective optimization. pages 462–470, 2013.